UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TRABAJO FINAL DE GRADO

Grado en Ingeniería Biomédica

# ANÁLISIS EN LAS DEFORMACIONES Y TRACCIONES DE UN EMBRIÓN DE MOSCA DROSOPHILA MELANOGASTER

Memoria y Anexos


**Autor:** Daniel Alcaraz Pérez
**Director:** Jose Muñoz Romero

**Convocatoria:** Mayo, 2018

# Resumen

Este trabajo surge de un estudio sobre las deformaciones y tracciones que experimentan las células del sistema nervioso central de un embrión de mosca *Drosophila melanogaster*. A cargo, José Muñoz, investigador del Laboratori de Càlcul Numèric (LaCàN) y profesor de la Universitat Politécnica de Catalunya. La técnica aplicada en este estudio es la Fast Iterative Digital Volume Correlation (FIDVC).

Por un lado se estudia el error generado a partir del método FIDVC en el algoritmo Particle Image Velocimetry (PIV). Por otro lado, se convierten los datos obtenidos al formato VTK para el software Paraview, un programa que parece adaptarse mejor a las exigencias del Post-Procesado.

# Resum

Aquest treball sorgeix d'un estudi sobre les deformacions i traccions que experimenten les cèl·lules del sistema nerviós central d'un embrió de mosca *Drosophilia melanogaster*. A càrrec, Jose Muñoz investigador del Laboratori de Càlcul Numèric (LaCàN) i professor de la Universitat Politècnica de Catalunya. La tècnica aplicada en aquest estudi es la Fast Iterative Digital Volume Correlation (FIDVC).

Per una banda s'estudia l'error generat a partir del mètode FIDVC en l'algoritme Particle Image Velocimetry (PIV). Per altra banda, es converteixen les dades obtingudes al format VTK per el software Paraview, un programa que sembla adaptar-se millor a les exigències del Post-Processat.

# Abstract

This project arises from the study on deformation and traction of CNS cells of a *Drosophila melanogaster* fly embryo. The study was conducted by Mr. José Muñoz, researcher at del Laboratori de Càlcul Numèric (LaCàN) and professor at UPC, with the application of Fast Iterative Digital Volume Correlation method (FIDVC).

On the one hand the FIDVC-led error, developed within the algorithm, is studied. On the other hand, generated data are converted into VTK arrays compatible with Paraview, a software that seems to better meet the requirements of Post-Processing.

# Agradecimientos

Agradecer este trabajo a mi tutor de TFG Jose Muñoz Romero, a mi familia y amigos por estar en los momentos difíciles del proyecto, ellos han estado apoyándome con fuerza en las horas de estudio.

# Glosario

*PIV: Particle Image Velocimetry*

*SNC: Sistema Nervioso Central*

*CNS: Nervous Central System*

*DIC: Digital Image Correlation*

*DVC: Digital Volume Correlation*

*FIDVC: Fast Iterative Digital Volume Correlation*

*PI: Procesamiento de imágenes*

*VTK: Visualization ToolKit*

*CLSM: Confocal laser scanning microscopy*

*ZNCC*: Zeroed Normalized Cross Correlation

*IDM: Image Deformation Method*

# Índice

# 1. Prefacio

El Procesado de Imágenes (PI) es una rama de la tecnología que tiene como objetivo modificar, estudiar y gestionar las imágenes obtenidas por dispositivos tecnológicos. Esta ciencia, cuyos orígenes se remontan al siglo pasado, está teniendo un papel importante en el nuevo modelo tecnológico que se quiere para los nuevos retos actuales, abriéndose camino en campos tan diversos como el de la inteligencia artificial, la biomedicina, la armamentística e incluso el arte. En este Trabajo de Final de Grado se describen técnicas de PI para el estudio biomecánico del Sistema Nervioso Central (SNC) de un embrión de mosca *Drosophila melanogaster*. Así, comprender el origen de algunas enfermedades y dolencias relacionadas con la médula espinal del ser humano.

## 1.1. Origen del trabajo

El trabajo tiene origen en la colaboración del Institut de Biologia Molecular de Barcelona (IBMB, CSIC), en particular el grupo de investigación liderado por Enrique Martín Blanco, junto con el Mechanobiology Institute de la National University of Singapore y el grupo de investigación LaCàN de la Universitat Politècnica de Catalunya.

## 1.2. Motivación

En estos años de formación en el grado de Ingeniería Biomédica se trabaja en las asignaturas de Matemáticas, Física, Química y Biología. Siendo éstas por las que he sentido un mayor interés, especialmente en ver como se combinaban entre ellas, en resumen, la aplicación de éstas en los fenómenos naturales. El desarrollo de este TFG ha sido un primer contacto con este mundo.

El análisis mecánico de los movimientos embrionarios es hoy en día necesario para la completa interpretación de los factores que gobiernan su desarrollo. Este trabajo estudia la aplicación de algoritmos que realizan el cálculo automático de los desplazamientos a partir de las imágenes de microscopía para el posterior análisis de las fuerzas que regulan la morfogénesis.

## 1.3. Requerimientos previos

Se necesitan conocimientos previos en Matlab y tener conocimientos en Procesamiento de Imágenes, ya que se tocan algunas técnicas como la correlación y filtrado de imágenes.

# 2. Introducción

Desarrollamos herramientas numéricas para entender las fuerzas que guían la morfogénesis. Combinando el análisis directo e inverso simulado el desarrollo embrionario del epitelio y el tejido.

## 2.1. Objetivos del trabajo

El objetivo de este trabajo se ha planteado bajo dos puntos centrales. Un primer punto, se basa en la adaptación de los datos del algoritmo PIV a un formato VTK para el programa de Post-Procesado Paraview. Y un segundo punto, está focalizado en el error generado por la técnica FIDVC en PIV.

## 2.2. Alcance del trabajo

El trabajo se divide en una parte teórica y una parte práctica. En la parte teórica se describen las técnicas de PI: Digital Image Correlation (DIC); Digital Volume Correlation (DVC); y Fast Iterative Digital Volume Correlation (FIDVC). También, se explica la teoría para entender el cambio de formato de los datos obtenidos a VTK. En la parte práctica se describe el algoritmo PIV, los bucles de acción y las modificaciones que se han llevado a cabo para el cálculo del error. También, generar una imagen del embrión de mosca *Drosophila melanogaster* en formato VTK.

# 3. Introducción a la Investigación

## 3.1. La mosca frutera Drosophila Melanogaster

La investigación "*Cell and Tissue Mechanics in Embryogenesis*" se centra en el estudio de los desplazamientos y tracciones que generan las células del Sistema Nervioso Central (SNC) en la mosca frutera *Drosophila melanogaster*, una especie que pertenece a la familia de las *Drosophilidae*. Utilizada frecuentemente en experimentación genética ya que mantiene un número reducido de cromosomas y un breve ciclo de vida. Además, aproximadamente el 61% de los genes que causan enfermedades en humanos tienen un homólogo en la mosca frutera. (1)

## 3.2. Datos obtenidos de la Investigación

El Institut de Biologia Molecular de Barcelona con la técnica Confocal Laser Scanning Microscopy (CLSM), obtiene las imágenes de la mosca frutera *Drosophila melanogaster*. En ellas (las imágenes), se observa la formación del SNC en el tiempo en varias capas de profundidad. FIJI, software de visulaización, genera un video en *2D* de todas las imágenes en los diferentes niveles de profundidad observados en el microscopio.

## 3.3. Confocal Laser Scanning Microscopy

El Confocal Laser Scanning Microscopy (CLSM) es un sistema óptico en el que se utiliza un láser como fuente de energía, un sistema de desviación y un sistema de lentes que guía la luz hacia un objetivo donde se encuentra el objeto a escanear, un control electrónico y un ordenador para el procesado de las imágenes. El objeto es escaneado punto por punto en diferentes profundidades y zonas.

Las partículas que reflejan la luz del Laser que incide en el sistema nervioso de la mosca son las proteínas FAS II que se adhieren a algunas células de la mosca, llegando a dar diversas tonalidades sobre la muestra [Figura 1]. (2)

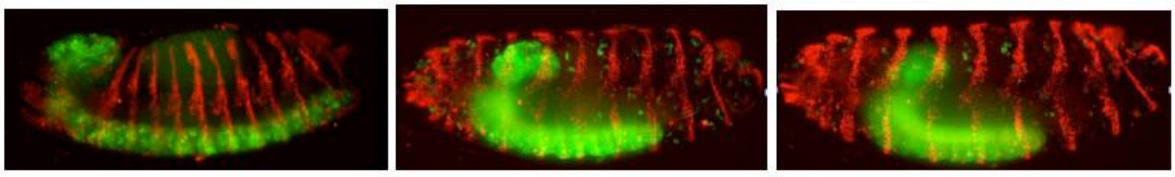


*Figura 1: Mosca frutera Drosophila melanogaster*

# 4. Técnicas de Procesado de Imágenes

En el Procesado de Imágenes hay una infinidad de técnicas para procesar las imágenes obtenidas de un dispositivo. Estas técnicas tienen como principal objetivo: El filtrado de imágenes, para eliminar ruido, suavizar o realzar alguna parte en concreto de la imagen; y el reconocimiento de objetos o la búsqueda de información. Para poder llevar a un terreno práctico estas técnicas se utilizan recursos matemáticos como por ejemplo la Transformada de Fourier, que permite separar o agrupar diferentes frecuencias en un mismo dominio.

En el código PIV de éste trabajo se utiliza la técnica FIDVC. Para entender ésta debe ser explicada la técnica Digital Image Correlation (DIC) y Digital Volume Correlation (DVC).

## 4.1. Digital Image Correlation

DIC es un método óptico que tiene como función determinar el movimiento de una imagen en el tiempo. Actualmente está siendo utilizada para el estudio del comportamiento de materiales, ya que es un método no invasivo y bastante económico. Tiene aplicaciones en muchos campos como la aeronáutica, la ingeniería de los materiales, la biomedicina, la ingeniería civil, la inteligencia artificial, etc…. Un ejemplo de la extensa variedad de opciones que presenta DIC es el trabajo que está realizando el National Physical Laboratory para paliar las emisiones de $CO_2$ en la industria a partir de materiales especiales capaces de captar esta molécula.

### 4.1.1. Correlación cruzada

La correlación de imágenes tiene como objetivo encontrar un campo de desplazamientos sobre un dominio en un intervalo de tiempo. Los investigadores Brue D. Lucas y Takeo Kanade en la Carnegie-Mellon University (2), describen con detalle esta técnica a partir de dos imágenes [Figura 2]. Un tamplate *G(x)* que se desplaza sobre *F(x)* con un vector *h.*

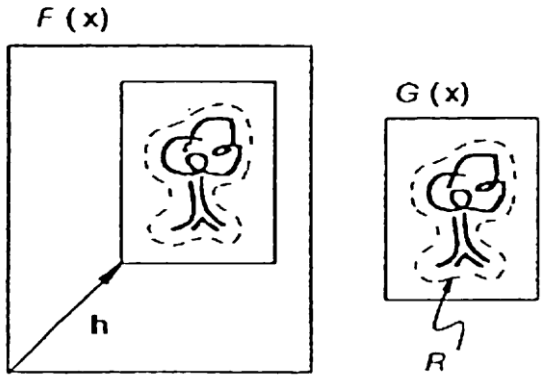



*Figura 2: Correlación entre una imagen F(x) y un template G(x)*

Las variables de entrada $Ia$ y $Ib$ son las imágenes a correlar, $Ia$ la imagen normal y $Ib$ la imagen deformada. En ellas se tienen los valores de intensidad de gris de cada pixel. Las imágenes $Ia$ y $Ib$ deben estar en escala de gris y no en *RGB*. La correlación cruzada tiene varias definiciones matemáticas, la *ZNCC* (Zeroed Normalaized Cross Correlation) que se utiliza en la investigación de Brue D. Lucas y Takeo Kanade [Ec.1] y es de gran ayuda para entender el proceso matemático. Pero aquí, se muestra la propuesta por J. Westerweel y C. Poelma de la Universidad Técnica de Delft [Ec.1]. (3)

$$R(i,j) = \frac{\sum_{k=1}^{Bx}\sum_{l=1}^{By}[Ia(k,l) - \overline{Ia}] * \left[Ib(k+i,l+j) - \overline{Ib}\right]}{\sqrt{\sum_{k=1}^{Bx}\sum_{l=1}^{By}[Ia(k,l) - \overline{Ia}]^2 * \sum_{k=1}^{Bx}\sum_{l=1}^{By}\left[Ib(k+i,l+j) - \overline{Ib}\right]^2}} \quad (1)$$

Donde los valores $Bx$ y $By$ son el tamaño del dominio, las dimensiones de la imagen en pixeles, en el *Eje x* y en el *Eje y*. El vector $\vec{h} = (i,j)$ es el que se encarga de desplazar los valores de la imagen $Ib$ sobre la imagen $Ia$ que se mantiene estática. Cuando se expresan los valores $Ia$ y $Ib$ , éstos son el valor en escala de la intensidad de gris de un pixel. Los valores $\overline{Ia}$ [Ec.2] y $\overline{Ib}$ [Ec.3] son la media de intensidad de la escala de gris en las dos imágenes, estos serán siempre constantes en la correlación $R(i,j)$.

$$\overline{Ia} = \frac{1}{BxBy}\sum_{k=1}^{Bx}\sum_{l=1}^{By} Ia(k,l) \quad (2)$$

$$\overline{Ib} = \frac{1}{BxBy}\sum_{k=1}^{Bx}\sum_{l=1}^{By} Ia(k,l) \quad (3)$$

Se expone un ejemplo para observar la correlación entre imágenes sobre un único eje, en un tiempo inicial $t_0$ y un tiempo final $t_f$ [Figura 3]. La correlación cruzada muestra en que punto del eje X las dos imágenes son más parecidas. El resultado es un coeficiente de intervalo *0* y *1*, siendo el *1* la máxima y *0* ninguna correlación.

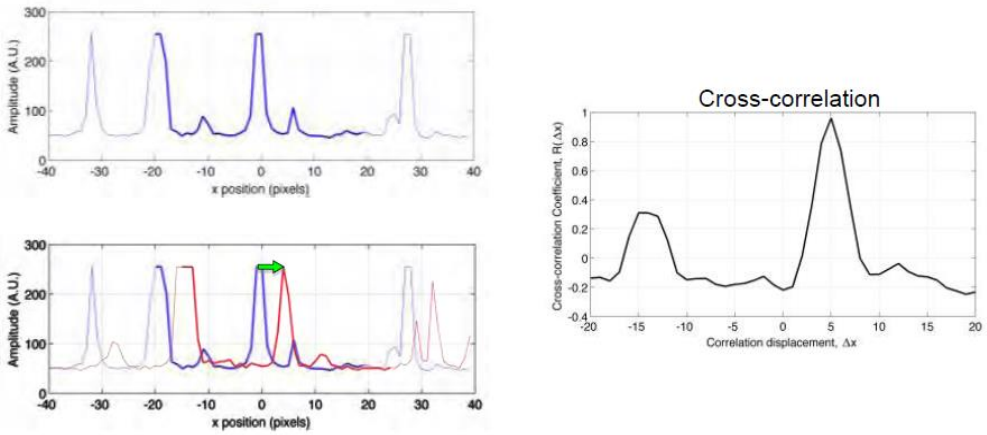



*Figura 3: Correlación Cruzada en DIC, en una dimensión a modo de representación gráfica*

La primera gráfica arriba a la izquierda muestra la representación de la imagen A. En el eje de abscisas se representa la variable *x* (Posición) y en el eje de ordenadas el valor de cada uno de los pixeles. En la segunda gráfica abajo a la izquierda se observa como la imagen A (trazo azul) es desplazada sobre una imagen B (trazo rojo) y se va produciendo en cada punto la correlación. El resultado es la gráfica de la derecha, que muestra con un pico en donde existe la correlación máxima entre las dos imágenes, en este caso cuando el desplazamiento es de 5 unidades respecto la imagen A se encontraría la máxima correlación, diríamos que la imagen se ha desplazado 5 unidades.

### 4.1.2. Errores en DIC

Existe un error real en el vector desplazamiento que se obtiene de calcular la correlación cruzada, éste depende de un valor *N*, es un error generado por las condiciones físicas en la captura de las imágenes [Ec.4].

$$N_I = \frac{C \cdot \Delta z_0}{M_0^2} \cdot D_I^2 \quad (4)$$

La $C$ es la concentración de partículas; El $\Delta z_0$ el espesor que tiene la cortina de luz; La $M_0$ la magnificación y el $D_I$ el tamaño del área de la partícula.

El promedio *N=10* es ideal para que el desplazamiento de pico sea el idóneo y no encontrar errores significantes en el vector desplazamiento. El error al hacer la DIC puede venir dado por la relación de este parámetro *N* con el número *e* [Ec.5].

$$Error\ PIV = \frac{e}{\sqrt{Ni}} \quad (5)$$

Se concluye que para encontrar una aproximación exacta del desplazamiento que realiza el dominio va a depender del tamaño de las partículas sobre las que se hace la correlación; el ruido de la adquisición de datos; y, la división de la proporción.

## 4.2. Digital Volume Correlation

DVC es un método que tiene como objetivo determinar el desplazamiento de una imagen que se va moviendo en un espacio de tres dimensiones durante un tiempo. Cada vez, son más las técnicas que se desarrollan para captar una imagen en 3D ya que los dispositivos y aparatos tienen una mayor capacidad tecnológica.

Para poder calcular los desplazamientos que se generan sobre un cuerpo en un tiempo, se utiliza un concepto nuevo en la correlación, el de *Sub-Volumen,* adaptado por los investigadores Bing Pan, Dfang Wa y Zhaoyang Wang, quienes describen el principio básico del DVC [Figura 4]. (4)

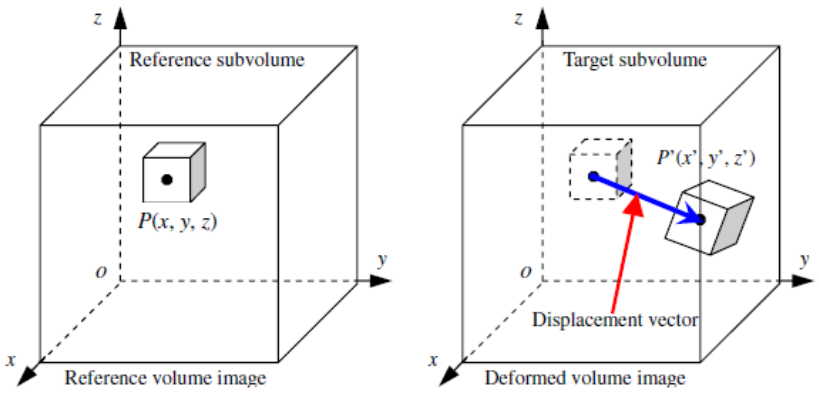



*Figura 4: Esquema Digital Volume Correlation*

Los pasos que siguen los investigadores para el desarrollo de la técnica DVC son los siguientes:

1. Adquisición de la imagen volumétrica.
2. Calcular el campo de desplazamiento. Especificar el punto de referencia de la imagen y rastrear posibles semejanzas.
3. Hacer una estimación de las tensiones. Calcular la tensión a partir del desplazamiento obtenido anteriormente.

Primero se determina el Sub-Volumen de referencia de dimensiones $sx\ \ sy\ \ sz$ en una región del espacio tridimensional centrado en un punto $p(p_x, p_y, p_z)$ que se irá desplazando conforme se van ejecutando las iteraciones de la correlación. El desplazamiento del punto central se realiza con los vectores $u_0, v_{0,} w_0$. La Zeroed Normalized Cross Correlation (ZNCC) es un coeficiente que indica la correlación entre la imagen no deformada *f(x)* y la imagen deformada *g(x)* [Ec.6].

$$C_{ZNCC}(u_0, v_{0,} w_0) = \frac{\sum_{i=1}^{n}[f(xi, yi, zi) - fm][g(xi + u_0, yi + v_0, zi + w_0) - gm]}{\sqrt{\sum_{i=1}^{n}[f(xi, yi, zi) - fm]^2} * \sqrt{\sum_{i=1}^{n}[g(xi + u_0, yi + v_0, zi + w_0) - gm]^2}} \quad (6)$$

El coeficiente $C_{ZNCC}$ tiene un resultado de intervalo [-1, 1]. El 1 tiene como valor la perfecta correlación, el 0 ningún tipo de correlación entre imágenes y el -1 la correlación opuesta. Las variables $fm$ y $gm$ son la intensidad media de cada uno de los Sub-Volumenes de la imagen no deformada y deformada [Ec.7][Ec.8].

$$fm = \frac{1}{n}\sum_{i=1}^{n}(xi, yi, zi) \quad (7)$$

$$gm = \frac{1}{n}\sum_{i=1}^{n} g(xi + u_0, yi + v_{0,}, zi + w_0) \quad (8)$$

### 4.2.1. Errores en Digital Volume Correlation

La intensidad lumínica varía dependiendo de la imagen de referencia, esto implica que el valor de intensidad de gris de los píxeles pueda aumentar o disminuir y, por tanto, la correlación no es del todo satisfactoria.

El sub-volumen parte de ser un cuerpo rígido, pero este durante el tiempo va a tener que ir cambiando de orientación y de forma, ya que los cuerpos que se deforman como el de la *Drosophila melanogaster* no lo hacen linealmente, sino que son irregulares.

### 4.2.2. Soluciones a los errores de Digital Volume Correlation

Para la intensidad lumínica que es variable en las imágenes, se añaden dos constantes *a* y *b* sobre una de las imágenes para buscar el equilibrio de intensidad de forma lineal [Ec.9].

$$g(xi', yi', zi') = a * f(xi, yi, zi) + b \quad (9)$$

Para el problema de las rotaciones, formas y orientaciones que puede adoptar en el tiempo el Sub-Volumen, se emplea una solución lineal en donde las nuevas coordenadas del este Sub-Volumen dependerán de la variable de entrada de tres vectores [Figura 5].

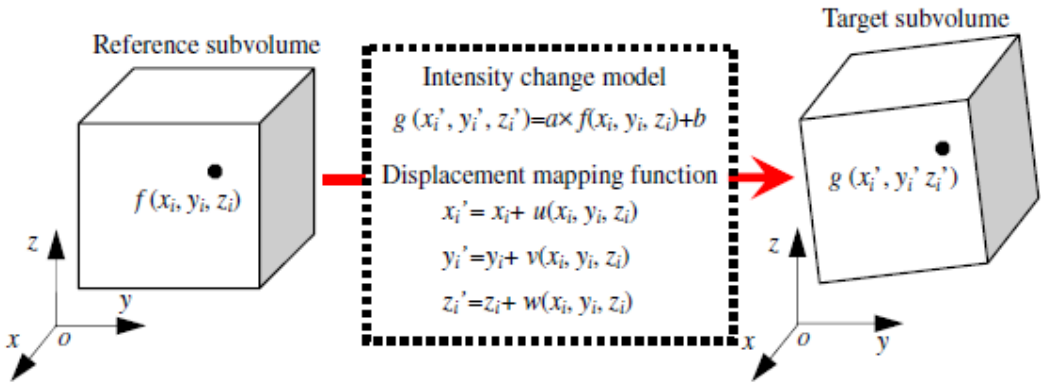



*Figura 5: Solución a los dos problemas de DVC: El cambio de intensidad en el modelo; y el desplazamiento de Sub-Volumen*

Las nuevas coordenadas del Sub-Volumen de la imagen deformada *g(x)* se definen por los tres vectores *u*, *v*, *w* [Ec.10]

$$x'_i = x_0 + \Delta x + u_0 + \Delta u + u_x\Delta + u_y\Delta + u_z\Delta$$

$$y'_i = y_0 + \Delta y + v_0 + \Delta v + v_x\Delta + v_y\Delta + v_z\Delta$$

$$z'_i = z_0 + \Delta z + w_0 + \Delta w + w_x\Delta + w_y\Delta + w_z\Delta \quad (10)$$

Los intervalos $\Delta x, \Delta y, \Delta z$ son la distancia entre el punto central $x_0,\ y_0,\ z_0$ en *f(x)* y el punto central en *g(x)* $x_i,\ y_i,\ z_i$. Ahora, queda la función *Fi(p)* con todos los variables obtenidas [Ec.11]

$$Fi(p) = g\begin{pmatrix} x_i + u_0 + +u_x\Delta x_i + u_y\Delta y_i + u_z\Delta z_{i,} \\ y_i + v_0 + +v_x\Delta x_i + v_y\Delta y_i + v_z\Delta z_{i,} \\ z_i + w_0 + +w_x\Delta x_i + w_y\Delta y_i + w_z\Delta z_i \end{pmatrix} - a \times f(x_i, y_i, z_i) - b \quad (11)$$

Esta función $Fi(p)$, con $p = \{\Delta u\ u_x\ u_y\ u_z\ \Delta v\ v_x\ v_y\ v_z\ \Delta w\ w_x\ w_y\ w_z\ a\ b\}^T$ y sabiendo que es una función no lineal, se puede transformar a la forma lineal con el coeficiente de primer orden de Taylor. Entonces queda la expresión en función de p, que es la variable que recoge todos los outputs [Ec.12] y la *k, que* es el contador de las iteraciones que se hacen en la correlación para encontrar el desplazamiento óptimo. Finalmente, la expresión de la Ecuación 12 se simplifica [Ec.13].

$$F_i(p^{k+1}) = F_i(p^k) + \nabla F_i(p^k)(p^{k+1} - p^k) \quad (12)$$

$$F_i(p^k) = g(x'^k_i, y'^k_i, z'^k_i) - a^k f((x_i, y_i, z_i) - b^k \quad (13)$$

### 4.2.3. Inconvenientes del Digital Volume Correlation

El coste computacional para realizar el DVC es muy elevado, se necesita mucho tiempo para obtener los valores, ya que entran en cada iteración *k* 14 variables. Esto supone una carga computacional importante en el cálculo en cada iteración que se realice.

## 4.3. Fast Iterative Digital Volume Correlation

El tiempo de ejecución que se emplea en el DVC es muy amplio, así como el número de iteraciones para encontrar el vector desplazamiento. Para reducir todo esto, se creó un algoritmo nuevo, el FIDVC. En el algoritmo PIV desarrollado por el profesor Jose Muñoz se utiliza de esta técnica. FIDVC se desarrolló en The Frank Laboratory, un laboratorio vinculado a la Universidad de Brown por los científicos E. Bar-Kochba, J. Toyjanova, E. Andrews, K.-S Kim y C.Frank (5) .

La adquisición de las imágenes para FIDVC se realiza vía Computed Tomography (CT), Magnetic Resonance Image (MRI) o Laser Scanning Confocal Microscopy (LSCM).

En FIDVC existen dos imágenes, la no deformada $I_0(x)$ y la deformada $\hat{I}_0(x)$, ellas se definen por tener dimensiones sobre los tres ejes del espacio. Algunas de las variables con las que se juegan es el espacio que hay entre celda y celda llamado *overlaps* y también se tiene la *interrogation window* que sería el Sub-Volumen con el que se hace la correlación. Estas dos variables (overlaps y interrogation window) son muy importantes para el cálculo de FIDVC.

La correlación se realiza a partir de un contador *k* que cuenta el nombre de iteraciones necesarias para obtener el valor del vector $u^k$ entre la imagen $I_0(x)$ y $\hat{I}_0(x)$. Existe un incremento en el desplazamiento para cada una de las iteraciones realizadas $du$ ($du_1$ $du_2$ $du_3$), los desplazamientos los obtenemos con la ecuación de correlación [Ec.14].

$$I^{k-1}(x) \otimes \hat{\mathrm{I}}^{k-1}(x) \rightarrow du \quad (14)$$

$$C(du) = \sum_{x=-\frac{L}{2}}^{\frac{L}{2}} f(x) * f'(x + du) \quad (15)$$

En la ecuación se muestra la forma que tiene una correlación entre dos imágenes, con el parámetro *k* que realiza las iteraciones. En la ecuación [Ec.15] se busca la correlación entre $f(x)$ y $f'(x+du)$, sobre valores de intensidad de nivel de gris que tienen cada una de las imágenes. Los intervalos $L/2$ es el tamaño de la *interrogation window*. FIDVC tiene una línea de acción acorde al siguiente diagrama de bloques [Figura 6]:

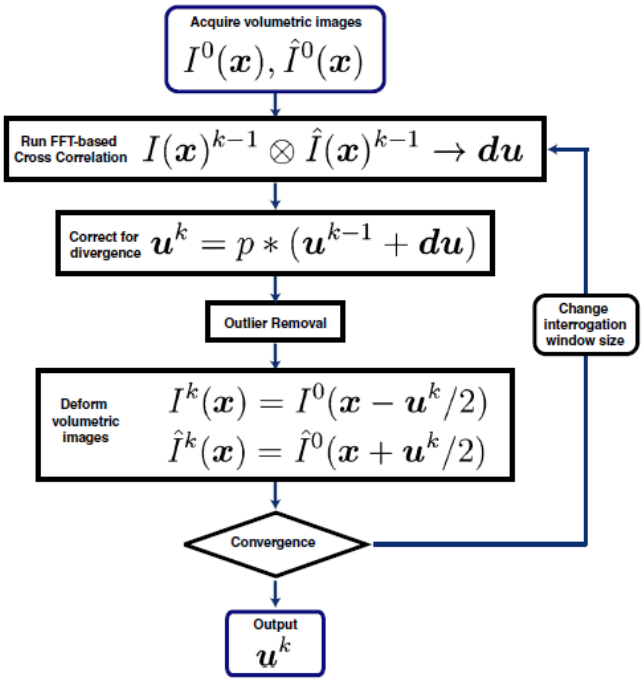



*Figura 6: Diagrama de bloques FIDVC*

En la correlación, algo que se debe tener en cuenta es la resolución de la ventana de interrogación, para mejorarla, se aplica un filtrado del tipo Nyquist. Este filtro, se aplica con una función llamada *welghting function w(x)*, que actúa dentro de la correlación [Ec.16].

$$w(x) = \prod_{i=1}^{3}(12\left|\frac{x_i}{L}\right|^2 - 12\left|\frac{x_i}{L}\right| + 3 + 0{,}15\cos(4\pi\, x_i/L\,) + 0{,}2\cos(6\pi x_i/L) + 0{,}1\cos(8\pi x_i/L) + 0{,}05\cos(10\pi x_i/L))\text{^}1/2 \qquad (16)$$

Dónde *x(x1 x2 x3)* son las posiciones en dónde se encuentra la imagen, sobre los intervalos definidos en la longitud que debe tener la ventana *L x L*. Finalmente, con la aplicación de este filtro la correlación queda de la siguiente forma [Ec.17]:

$$C(du) = \sum_{x=-\frac{L}{2}}^{\frac{L}{2}} w(x)f(x) * w(x+du)\, f'(x+du) \qquad (17)$$

Esta función implementada en un espacio de Fourier quedaría de la siguiente forma [Ec.18] Esto se hace para visualizar la correlación en un mismo dominio.

$$C(du) = F^{-1}\{F\{w(x)f(x)\} * F\{w(x+du)f'(x+du)\}\} \qquad (18)$$

Todo y con la aplicación del filtrado de Nyquist no se elimina del todo el comportamiento bajo de la correlación.

El principio básico de comportamiento que tiene FIDVC se muestra en el esquema siguiente [Figura 7]. El modelo que se sigue en el esquema es el de *Image Deformation Method (IDM)*. La imagen normal y la deformada van rotando entre ellas hasta encontrarse. En cada iteración que se hace *k*, las dos imágenes se deforman simétricamente "*symetrically warped*".

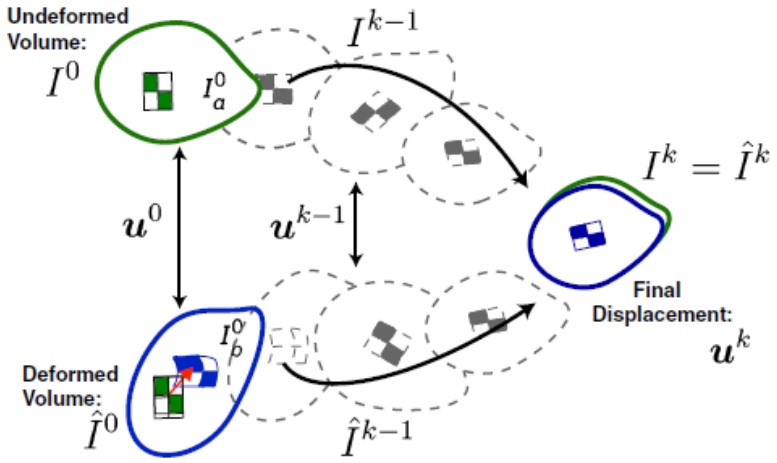



*Figura 7: Representación de las iteraciones en FIDVC*

La deformación entre la imagen normal y la deformada se realiza a partir del vector $u^k$ [Ec.19]. Que tiene una ecuación que se va implementando sobre las dos imágenes $I_0(x)$ y $\hat{I}_0(x)$ [Ec.20]. Cada imagen será en cada iteración distinta después de ser aplicado el factor $I^k$ y $\hat{\mathrm{I}}^k$ .

$$u^k = \sum_k u^{k-1} du \quad (19)$$

$$I^k = I^0\left(x - \frac{u^k}{2}\right) \qquad \hat{\mathrm{I}}^k = \hat{\mathrm{I}}^0\left(x + \frac{u^k}{2}\right) \quad (20)$$

IDM es algo inestable y se tienen algunos problemas con el vector de desplazamiento obtenido, no acaba de ser del todo bueno, por ello se aplica un filtrado tipo *p* [Ec.21] propuesto por Schrijer, éste se aplica en el vector de la siguiente forma [Ec.22]:

$$p(\xi) = \left(\frac{L}{2}\right)^z - |\xi|^z \qquad (21)$$

Donde $L$ es la longitud de la ventana. El parámetro *z* controla la fuerza del filtrado paso bajo.

$$u^k = p * \left(u^{k-1} + du\right) \quad (22)$$

A la pregunta ¿Cuándo se debe parar el número de iteraciones que se realizan en el algoritmo? Esto depende de el parámetro $e^k$ [Ec.23] donde se tiene en cuenta el valor de intensidad de gris de la imagen no deformada y deformada, $\sigma$ la media estándar de desviación y $N$ el número de voxels. En el esquema siguiente se muestra un diagrama de bloques de la técnica empleada [Figura 8].

$$e^k = \frac{1}{\sigma}\sqrt{\frac{1}{N}\sum_{x=1}^{N}\left(I^k(x) - \hat{I}^k(x)\right)^2} \qquad (23)$$

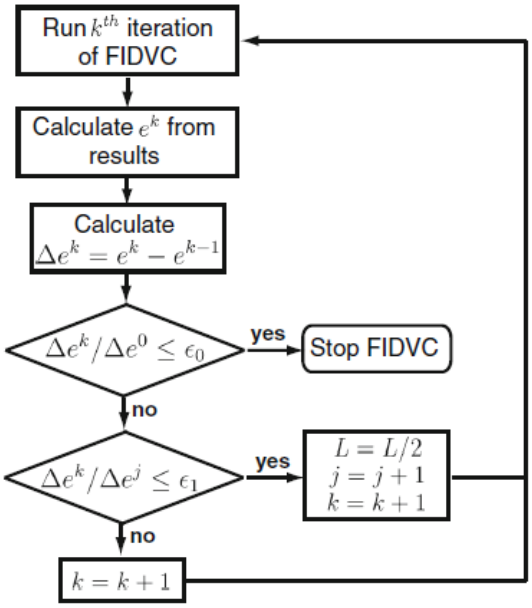



*Figura 8: Iteraciones y funcionamiento en FIDVC*

Cuando el bucle para "*STOP FIDVC",* se obtiene el valor desplazamiento y el número de iteraciones *k* que han hecho falta.

### 4.3.1. Inconvenientes Fast Irerative Digital Volume Correlation

El problema que puede presentar FIDVC el uso de los filtros, se tiene que pensar de forma experimental cuáles son los parámetros adecuados a introducir y así obtener un buen resultado.

# 5. Paraview y VTK

Los datos resultado que se han obtenido en PIV se ordenan en el algoritmo para visualizarlos en un programa de Post-Procesado. El programa de Post-Procesado que se utilizaba en la versión del algoritmo PIV es GID, no obstante, se ha querido utilizar otro tipo de software que parecía adaptarse mejor a las condiciones que se requieren en la técnica FIDVC. El programa es Paraview, diseñado para procesar datasets de gran tamaño. Paraview es un referente en análisis de datos. VTK "*Visualization ToolKit*" es el formato necesario para observar los resultados obtenidos en Paraview.

## 5.1. Paraview

Desarrollado en el año 2000 en el Alamos National Laboratory siendo desde un principio una plataforma abierta para los usuarios. Se puede descargar la última versión en la página web oficial www.paraview.org .

Permite la visualización de vectores y la organización de estructuras con diferentes bases geométricas: cubos; polígonos; polígonos sin estructura; hexágonos; etc. El uso de diferentes bases es útil y permite construir mallas o cuerpos con un cierto grado de libertad.

Un punto interesante es el filtrado de datos que incorpora, pudiendo filtrar, modificar o eliminar aquellos datos que no interesan.

## 5.2. VTK

VTK es un software libre editado en 1993 que genera gráficos 3D para ordenador y procesado de imágenes. En estos años la empresa que editó VTK, Kitware, se ha convertido en una de las grandes empresas referentes en la visualización 3D. VTK también dispone de varias plataformas de ayuda e instrucciones para organizar los datasets y, que éstos sean compatibles en los programas de Post-Procesado.

# 6. Práctica

En esta parte del TFG se expone como se han alcanzado los dos objetivos planteados en un principio, cambiar el formato de los "*datasets*" obtenidos en PIV a VTK y calcular el error existente en el código PIV *versión 11* (PIV.v11).

En la primera parte se explica el funcionamiento que presenta PIV por medio de un diagrama de bloques, para observar de forma gráfica las relaciones establecidas entre las diferentes partes del código PIV. Y, la explicación de algunas de las funciones como *PIV_CNS* y *GenerateMesh*.

En la segunda parte se explica el paso de los *Tifs* a VTK con la función *Tiffs2VTK*, como es su funcionamiento, explicación de algunos conceptos teóricos acerca de VTK (cómo se organizan los datos) y cómo la función se implementa dentro del programa PIV.*v11*. También en esta misma dirección, se estudia otra función creada, *VTKPIV* responsable de traducir a formato VTK el resultado desplazamiento y tracción en el programa general.

Finalmente, el cálculo del error que existe en el PIV.*v11* con la función *PIVError*, qué parámetros se han utilizado, explicación de la función *ADI* y *DisplacementImage*. Y, cómo la función *PIVError* se implementa dentro de PIV.*v11*.

Se puede decir que la modificación del algoritmo antiguo PIV.v11 da lugar a un nuevo programa que será la versión número 12, la cual tendrá los apartados nuevos que se han ido integrando.

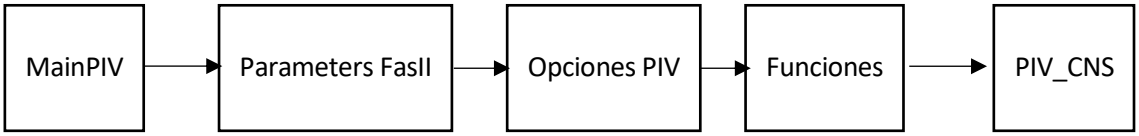
El material necesario para la parte práctica son: El programa de programación Matlab; El programa de Post-Procesado GID; El programa de Post-Procesado Paraview; El programa de visualización de imágenes FIJI; y, Los datos obtenidos de la mosca frutera *Drosophila melanogaster*.

Los datos que se han obtenido del sistema central de la mosca son del tipo *Tif* y tienen dimensiones *468 x 240* y un número total de 34 capas [Figura 9]. En total es un conjunto de 1394 imágenes que han sido recogidas en 40 unidades de tiempo.


*Figura 9: Imágenes Tiff de la mosca frutera Drosophila melanogaster en FIJI*

## 6.1. Código PIV versión 11

El código PIV.v11 consta de varias partes, para explicar éste es oportuno crear un diagrama de bloques en el que se vea un resumen de la línea de actuación que tiene. El diagrama de bloques se ha creado de la siguiente manera para que la explicación de éste sea más intuitiva, la parte técnica se encuentra explicada en cada paso:




***Main PIV***: Menú principal del algoritmo PIV

***Parameters FasII***: Configuración de los parámetros sobre las imágenes *Tif* que van a ser utilizadas para el cálculo PIV. Los parámetros utilizados son: El tamaño de la imagen en píxeles, en las tres dimensiones *xpixel, ypixel* y *zpixel*; Las dimensiones reales de la imagen *Lx, Ly* y *Lz*; El tamaño del sub-volumen para el FIDVC *sx, sy* y *sz;* El tiempo total *tmax*; y, Las tolerancias para el filtrado.

***Opciones PIV***: En este nivel, se presentan las opciones que tiene PIV. A continuación, se muestran punto por punto las opciones escritas que son variables booleanas.

1- *WriteGIDImages*: Escribir las imágenes *.tif* para el programa de Post-Procesado, GID.
2- *WriteVTKImages:* Escribir las imágenes *.tif* para el programa de Post-Procesado, Paraview.
3- *RotateImages*: Rotar las imágenes de la carpeta *tif*.
4- *WriteRotatedImages*: Escribir las imágenes rotadas.
5- *PIV*: Escoger el tipo de correlación que se quiere hacer para obtener los resultados del desplazamiento y las tracciones generadas en un dominio. El valor PIV=1 es la opción de desplazamiento Euleriano utilizando la función *xcorr*. Y, el valor PIV=2 utiliza la técnica óptica FIDVC.
6- *Set.Inverse*: Completar las opciones de desplazamiento.
7- *Set.Lagrangian*: Completar las opciones de tracciones en un dominio sobre un cuerpo que se va deformando.
8- *Set.Plot*: Gráfico de los distintos resultados obtenidos.

***Funciones:*** En esta parte se aplican con sus respectivas funciones las opciones que se han especificado al principio del código. Por ejemplo, si la opción *WriteVTKImages* es cierta, entonces hay una condición que lleva a ejecutar la función *Tifs2VTK* [Figura 10].

```
91 -    if WriteVTKImages
92 -       Tifs2VTK(name,Lx,Ly,Lz,tmax,xpixel,ypixel,zpixel);
93 -    end
```

*Figura 10: Ejemplo si la opción WriteVTKImages es cierta*

***PIV_CNS***: Eje central del código, sobre el que se obtienen los resultados, en *PIV_CNS* se aplican las técnicas ópticas de FIDVC y Euleriana. A continuación, se explica por pasos cómo funciona esta:

Primer paso: Coger las imágenes de una carpeta, seleccionar los nodos $nx\ ny\ nz$ que existen en función del tamaño de las imágenes y del Sub-Volumen de referencia de dimensiones $sx\ sy\ sz$. Los nodos indican aquellos puntos de la imagen en donde se concentra la información se obtiene, de vectores desplazamiento y valor intensidad de gris.

Segundo paso: Generar una malla "*Mesh",* la función encargada de ésto es *GenerateMesh* que escribe una matriz $X$ con todos los puntos y otra matriz $C$ con las celdas, éstas dos matrices sostienen las imágenes para darles una forma 3D. A continuación, se introducen valores característicos cómo las tolerancias, que filtran puntos o desplazamientos que no son de interés.

Tercer paso: Aplicar PIV, hay dos opciones, o bien que se trabaje con los desplazamientos Eulerianos, en ese caso se correría la función *funXcorr3*; o que se trabaje con la técnica de procesado FIDVC y, en ese caso se corre la función *funIDVC2*. Al aplicar PIV se obtienen tres vectores $u_x\ u_y\ u_z$ que indican el desplazamiento, la función *Grid2Mat* permite agrupar estos tres para obtener un único vector desplazamiento $u$.

Cuarto paso: Si, *Set.Inverse=true* entonces el programa calcula las tracciones a partir del análisis inverso con la función *InverseFEM*, algo importante de esta función son las máscaras que se aplican a los vectores desplazamientos, dependiendo del valor que tengan.

Quinto paso: Pasar los vectores desplazamiento y de tracción a formato VTK para que de esta forma se puedan observar en Paraview. Se utiliza para ello la función *VTKPIV*.

### 6.1.1. Tabla de variables de entrada y salida en las funciones de PIV_CNS

| Función | Variables de entrada | Variables de salida |
|---|---|---|
| PIV_CNS | Name; Lx; Ly; Lz; Opt; Opt.ov; Opt.sx sy sz; pSizes= [xpixel ypixel zpixel] | ux( i, j, k, t); uy( i, j, k, t); uz( i, j, k, t); c( i, j, k, t); |
| SetSizes | pSizes; Opt; Set | sSizes; ov |
| ReadImages | Folder; tmin; zpixel | Image2 |
| SetnSizes | Set.dm; image2; L; pSizes; sSizes; Set.PIV | D; nx; ny; nz; oSizes |
| GenerateMesh | D(1); D(2); D(3); nx-1; ny-1; nz-1 | C; X |
| funXcorr3 | Image1; image2; nSizes; oSizes; pSizes; Set; t; tmax; | ux( i, j, k, t); uy( i, j, k, t); uz( i, j, k, t); c( i, j, k, t); cI; |
| funIDVC2 | Set.dm; image1; image2; sSize; t; tmax | ux( i, j, k, t); uy( i, j, k, t); uz( i, j, k, t); c( i, j, k, t); cI; |
| Grid2Mat | ux ( i, j, k, t); uy ( i, j, k, t); uz ( i, j, k, t); | u( : , : , t ) |
| MaskCU | Grid2Mat (c ( : , : , : , t )); Grid2Mat (cI); cmax; u( : , : , t ); Set.TolC; Set.MTol; | cMask ( : ,t ); mMask; u0Mask ( : ,t ); |
| InverseFEM | Set; C; X; u0; uD; eMask; mMask; tMask( : , t ); uMask( : , t ) | tractions; u ( : , : , t ); C |
| Dof2Mat | Set.dim; Set.nodes; tractions | tM ( : , : , t ); tMask( : , t ); |
| VTKPIV | Name; c; C; cMask; tM; tMask; tmin; t; u; uMask; u0Mask; u0MaskL; x; X; Set.Lagrangian | uMaskL |

*Tabla 1: Inputs & Outputs PIV_CNS*

## 6.2. Imagen 3D a VTK

Los archivos que se generan en la versión 11 de PIV están destinados al programa de Post-Procesado GID. *Tifs2VTK* permite que el algoritmo general muestre los archivos en formato VTK para el software Paraview. Esta función se diseña con la ayuda de un archivo de instrucciones de VTK que indica la manera de distribuir los diferentes elementos para que se puedan leer en un programa de Post-Procesado .

En cada tiempo se genera un archivo VTK, a diferencia de GID que para un experimento engloba todos los tiempos en un mimso archivo.

### 6.2.1. Función

Primer paso: Obtener los valores característicos de las imágenes, se tienen en cuenta los píxeles, los nodos y la separación que hay entre ellos conociendo el dominio real de las imágenes. Seguidamente, se aplica la función *GenerateMesh* que organiza los puntos de una imagen 3D en dos matrices: *X* que distribuye la organización de los nodos y *C* que es la matriz que designa las celdas.

Segundo paso: Conocer el valor de intensidad de gris de cada uno de los nodos en las diferentes capas de profundidad de la imagen 3D es importante. Para ello, la función *AssignGrayLevel* asigna el nivel de gris a cada nodo y los guarda en una matriz *GrayLevel* con dimensiones dadas.

Tercer paso: Escribir la información obtenida en formato VTK. Las instrucciones de VTK muestra en una tabla los tipos de malla. PIV utiliza la *Unstructured Grid* que permite combinar diferentes tipos de celda [Figura 11]. Las celdas que se utilizan para construir la malla son del *tipo 12* con forma de hexaedro irregular con ocho vértices (ocho vértices de conexión).

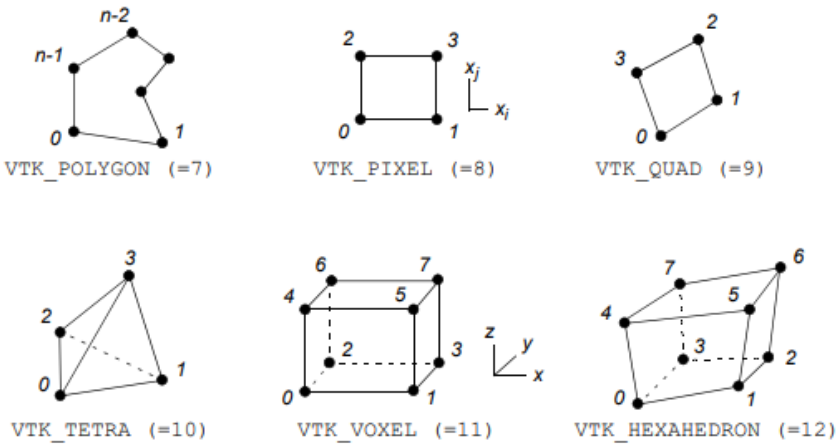



*Figura 11: Tablas tipos de celda*

En el encabezado de un documento VTK se escribe información acerca de la versión del software; el título del documento; el lenguaje ASCII (American Standars Code of Information Interchange), código basado en el alfabeto latín [Figura 12].

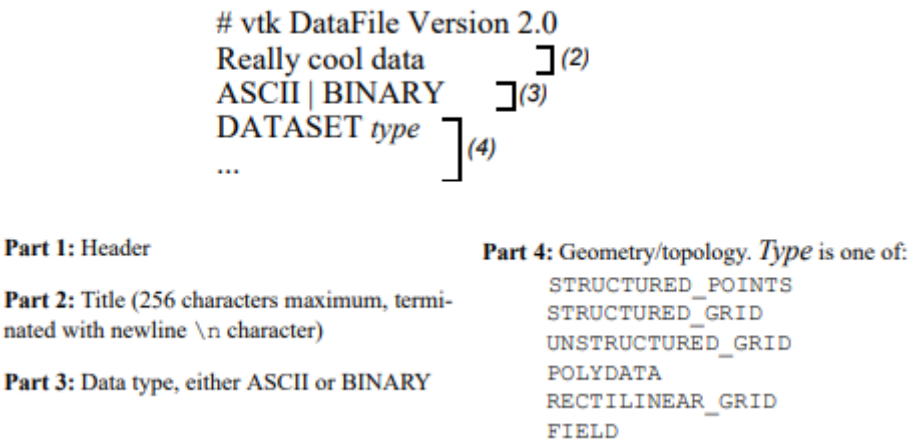



*Figura 12: Encabezado del documento VTK*

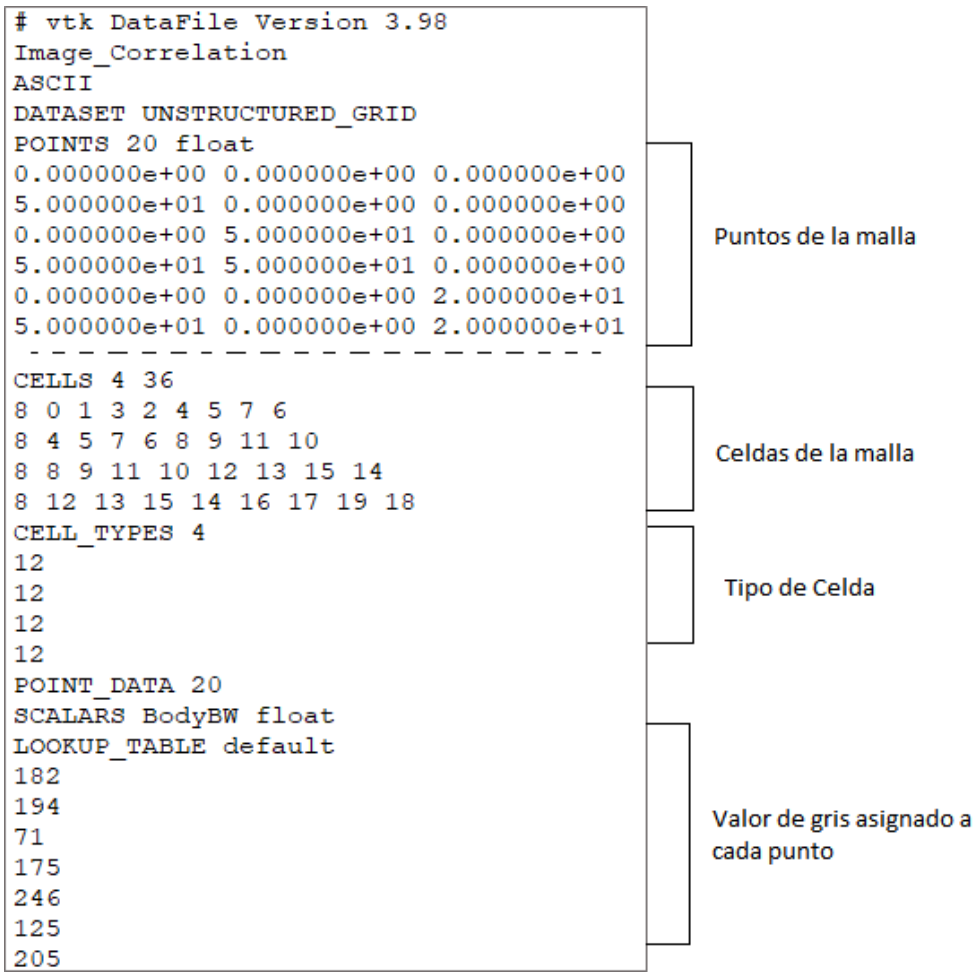
Al escribir los datos sobre la malla, los datos más importantes son: el número total de puntos y celdas. Con ellos (número de puntos y celdas) se escribe el VTK en forma de *vectores* o *escalares*. Los vectores son matrices de *i* filas y *j* columnas, pero los escalares son únicamente números colocados en una única columna, en la siguiente figura se muestra la disposición de éstos [Figura 13]. En VTK para escribir la imagen se utilizan *vectores* para los puntos y las celdas y, *escalares* para el nivel de intensidad de gris.

```
83      % WRITE VTK FILE
84 -     fid=fopen(strcat('VTK','t',num2str(t),'.vtk'),'w');
85 -     fprintf(fid,'# vtk DataFile Version 3.98\n');
86 -     fprintf(fid,'Image_Correlation\n');
87 -     fprintf(fid,'ASCII\n');
88 -     fprintf(fid,'DATASET UNSTRUCTURED_GRID\n');
89      % Write nodal coordinates
90 -     fprintf(fid,'POINTS %i float\n',npoints);
91 -     for n=1:npoints
92 -         fprintf(fid,'%e %e %e \n',X(n,:));
93 -     end
94      % Write elemental connectivity
95 -   fprintf(fid,'CELLS %i %i\n',ncells,ncells*nnodes);
96 -   for e=1:ncells
97 -        fprintf(fid,'%i ',nnodes-1,C(e,1:8)-1);
98 -        fprintf(fid,'\n');
99 -   end
100 -  fprintf(fid,'CELL_TYPES %i\n',ncells);
101 -  for e=1:ncells
102 -      fprintf(fid,'%i\n',12);
103 -  end
```

*Figura 13: Vectores y escalares VTK*

### 6.2.2. Formato

Este es el documento resultado generado por *Tifs2VTK* para una imagen generada con la función *ImageGenerator* [Figura 14].




```
# vtk DataFile Version 3.98
Image_Correlation
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 20 float
0.000000e+00 0.000000e+00 0.000000e+00
5.000000e+01 0.000000e+00 0.000000e+00
0.000000e+00 5.000000e+01 0.000000e+00
5.000000e+01 5.000000e+01 0.000000e+00
0.000000e+00 0.000000e+00 2.000000e+01
5.000000e+01 0.000000e+00 2.000000e+01
 - - - - - - - - - - - - - - - - - - - -
CELLS 4 36
8 0 1 3 2 4 5 7 6
8 4 5 7 6 8 9 11 10
8 8 9 11 10 12 13 15 14
8 12 13 15 14 16 17 19 18
CELL_TYPES 4
12
12
12
12
POINT_DATA 20
SCALARS BodyBW float
LOOKUP_TABLE default
182
194
71
175
246
125
205
```

*Figura 14: Visualización del formato VTK de la función Tifs2VTK*

### 6.2.3. Tabla de variables de entrada y salida en las funciones de Tifs2VTK

| Función | Variables de entrada | Variables de salida |
|---|---|---|
| Tifs2VTK | Name; Lx; Ly; Lz; tmin; tmax; xpixel; ypixel; zpixel; trim; Set.TolC | Format VTK |
| GenerateMesh | D(1); D(2); D(3); nx-1; ny-1; nz-1 | C; X |
| AssignGrayLevel | Image; GrayLevel; z; nx; ny | GrayLevel |

*Tabla 2: Inputs & Outputs Tifs2VTK*

### 6.2.4. Adaptación de la función al algoritmo PIV

En *MainPIV* con *WriteVTKImages* de carácter booleano, se ejecuta *Tifs2VTK*.

### 6.2.5. Resultados

Ejemplos de cómo funciona esta función con la ayuda de una función capaz de generar una imagen en 3D, se pasa esta imagen obtenida a un formato VTK y después se muestra en el programa Paraview.

#### 6.2.5.1. Resultado 1

Para el cáluclo del error que genera FIDVC en el algoritmo PIV se va a crear cubo, con la función *ADI,* de dimensiones cuadradas, *50x50x50* en los tres ejes del espacio *xyz* [Figura 15]. En este cubo hay una marca que se va moviendo en el tiempo con un desplazamiento de *u= [ 5 5 0 ]* . La representación de ésta imagen en Paraview es con Point Gausian, con un radio de esfera de 1.4. Tamvién, se muestran los ejes xyy con las dimensiones establecidas.

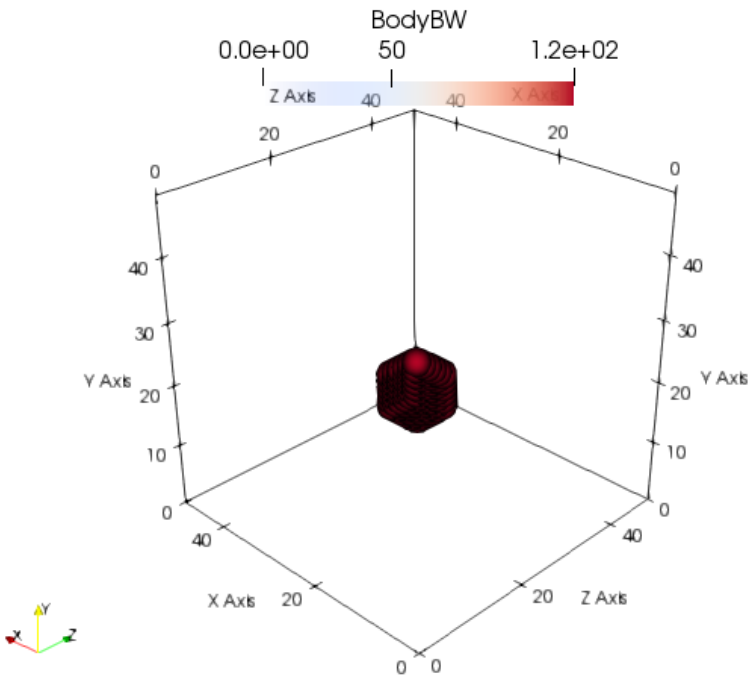



*Figura 15: Resultado 1*

#### 6.2.5.2. Resultado 2

Con ayuda de la función *TrimImages* se recortan parte de los tifs del sistema nervioso de la mosca *Drosophila melanogaster*. Esto es lo que se obtiene utilizando la función *Tifs2VTK*. La visualización en Paraview es con Point Gaussian con un radio de esfera de 1.4 [Figura 16].

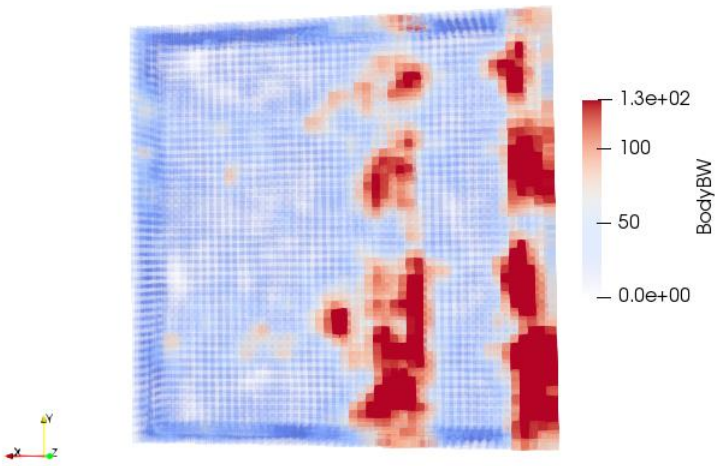



*Figura 16: Resultado 2*

#### 6.2.5.3. Resultado 3

El último resultado muestra la mosca *Drosophila melanogaster* en todo su dominio [Figura 17].

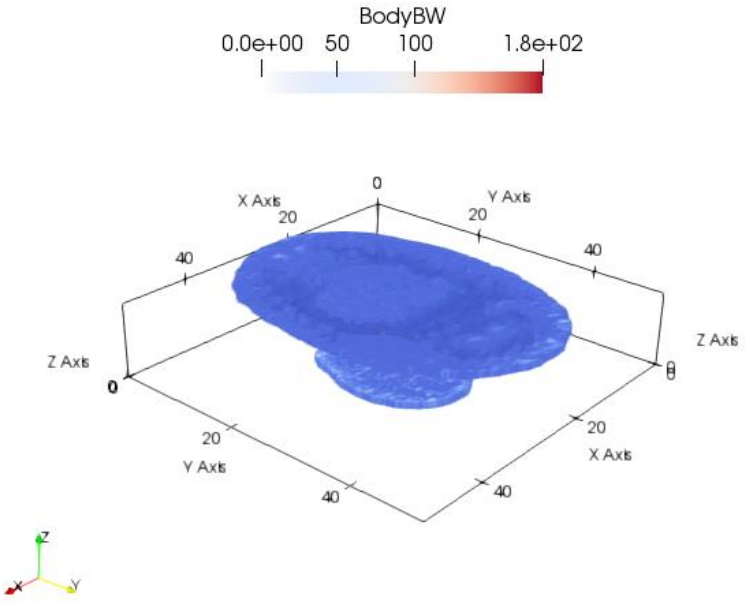



*Figura 17: Resultado 3*

## 6.3. Resultados de PIV a VTK

Los resultados de PIV a VTK se realizan con *VTKPIV*. En la creación de la función se ha utilizado como referencia la función *GID2VTK*, así como las instrucciones de VTK. *VTKPIV* se conjunta con dos funciones: *VTKMesh* y *VTKRes*. Al escribir los datos VTK se necesita siempre de una malla y, *VTKMesh* la escribe. *VTKRes*, escribe los resultados del desplazamiento y las tracciones. Hay que recordar que VTK genera un archivo para cada tiempo.

### 6.3.1. Función


***VTKPIV*:** En esta función entran los resultados de PIV. A continuación, se explica paso a paso el proceso que sigue:

Primer paso: Se abren los archivos en los que se introducen los resultados con formato VTK. Existen varios tipos de archivos dependiendo de la configuración que se introduzca al inicio del programa. En el nuevo algoritmo, sin embargo, se implementa sólo el archivo *VTKFileS*, los demás restan parados.

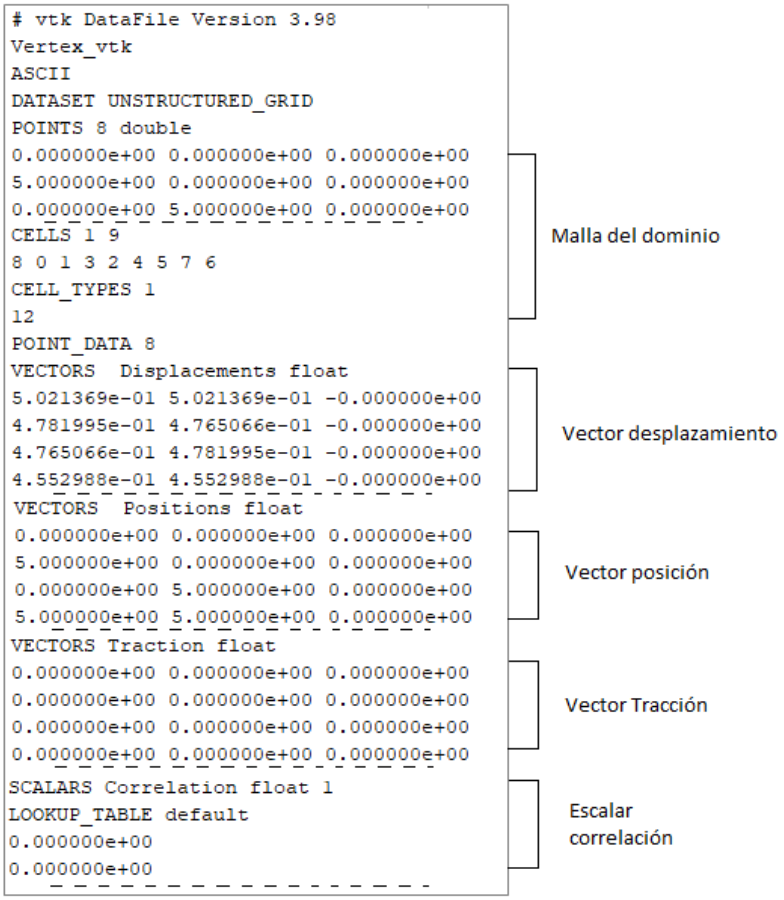
Segundo paso: *VTKMesh* escribe la malla que implementa a los valores de los resultados. Para futuros escenarios, existirá una alternativa para escribir la malla en Lagrangiano, es por esta razón que existe una condición del tipo *if*.

Tercer paso: Escribir los resultados obtenido de PIV_CNS en el archivo *VTKFileS* con la función *VTKRes*. En un escenario futuro, los archivos dependiendo de los parámetros introducidos inicialmente; los resultados se escribirán de diferente forma, como la utilización del concepto Lagrangiano.

Cuarto paso: Se cierran los archivos creados con la orden *fclose.* En el código *fclose(VTKFileS).*

### 6.3.2. Formato

Este es el documento resultado de la función *VTKPIV* para una imagen creada por la función *ADI* adjuntada en *MainPIV*. Imagen de dimensiones*, xpixel=5*, *ypixel=5* y *zpixel=3* con un desplazamiento *u= [1 1 0]*. [Figura 18]




```
# vtk DataFile Version 3.98
Vertex_vtk
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 8 double
0.000000e+00 0.000000e+00 0.000000e+00
5.000000e+00 0.000000e+00 0.000000e+00
0.000000e+00 5.000000e+00 0.000000e+00
CELLS 1 9
8 0 1 3 2 4 5 7 6
CELL_TYPES 1
12
POINT_DATA 8
VECTORS  Displacements float
5.021369e-01 5.021369e-01 -0.000000e+00
4.781995e-01 4.765066e-01 -0.000000e+00
4.765066e-01 4.781995e-01 -0.000000e+00
4.552988e-01 4.552988e-01 -0.000000e+00
VECTORS  Positions float
0.000000e+00 0.000000e+00 0.000000e+00
5.000000e+00 0.000000e+00 0.000000e+00
0.000000e+00 5.000000e+00 0.000000e+00
5.000000e+00 5.000000e+00 0.000000e+00
VECTORS Traction float
0.000000e+00 0.000000e+00 0.000000e+00
0.000000e+00 0.000000e+00 0.000000e+00
0.000000e+00 0.000000e+00 0.000000e+00
0.000000e+00 0.000000e+00 0.000000e+00
SCALARS Correlation float 1
LOOKUP_TABLE default
0.000000e+00
0.000000e+00
```

*Figura 18: Formato archivo VTK generado con VTKPIV*

### 6.3.3. Tabla de variables de entrada y salida en las funciones de VTKPIV

| Función | Variables de entrada | Variables de salida |
|---|---|---|
| VTKPIV | Name; C; c; cMask; tM; tMask; tmin; t; u; uMask; u0Mask; u0MaskL; x; X; Set.Lagrangian | Format VTK |
| VTKMesh | C; X; VTKfidS; uMaskL | Format VTK |
| VTKRes | VTKfid; New; time; X; c; cMask; t; tMask; u; uMask; u0; u0Mask | Format VTK |

*Tabla 3: Inputs & Outputs VTKPIV*

### 6.3.4. Adaptación de la función al algoritmo PIV

La adaptación de *PIVVTK* al algoritmo parte del eje vertebrador *PIV_CNS*, cuando ya se obtienen los resultado desplazamiento y tracción para un tiempo, se aplica la función *PIVVTK* [Figura 19].

```
154        % Write at each increment (get at least partial results)
155 -      uMaskL=VTKPIV(name,C,c,cMask,tM,tMask,tmin,t,u,uMask,u0Mask,u0MaskL,x,X,Set.Lagrangian);
```

*Figura 19: Línea VTKPIV en PIV_CNS*

### 6.3.5. Resultados

En estos resultados se muestra el vector desplazamiento en diversos casos sobre la imagen desplazada. Son un total de tres resultados. El primer resultado es el desplazamiento sobre una imagen virtual que ha sido desplazada con la función *ADI* y las otros dos resultados son sobre el desplazamiento que ejerce el Sistema Nervioso de la mosca *Drosophila melanogaster*.

El resultado se observa sobre el programa de Post-Procesado Paraview y para observar los vectores se utiliza una herramienta llamada *Glyph* que te permite ajustar éstos. En esta opción se observa la dirección y el sentido de los vectores, pero no la magnitud.

#### 6.3.5.1. Resultado 1

El cáluclo del error que genera FIDVC en el algoritmo PIV se va a crear un cubo con la función *ADI* de dimensiones cuadradas, *50x50x50* en los tres ejes del espacio *xyz* [Figura 18]. En este cubo hay una marca que se va moviendo en el tiempo con un desplazamiento de *u= [ 5 5 0 ]* [Figura 20].

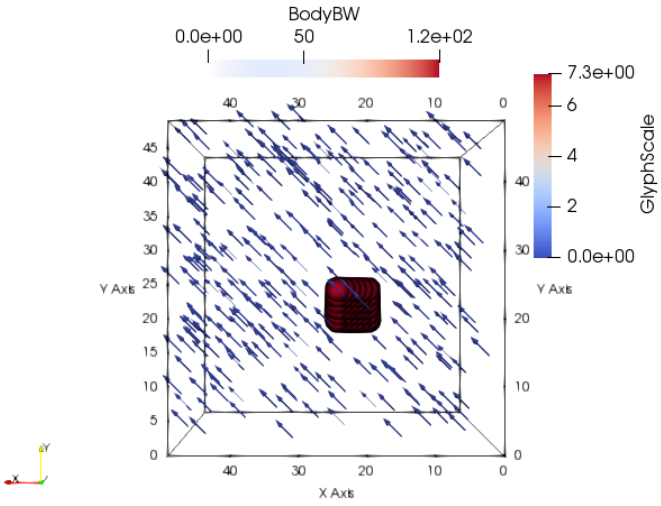



*Figura 20: Resultado 1 VTKPIV*

### 6.3.5.2. Resultado 2

Con ayuda de la función *TrimImages* se recorta parte de los tifs del sistema nervioso de la mosca *Drosophila melanogaster.* Esto es lo que se obtiene utilizando la función *Tifs2VTK*. La visualización en Paraview es con *Puntos* e implementado con el desplazamiento con la herramienta Glyph [Figura 21].

```
>> TrimImages('\Users\Daniel\Documents\Deformation_Analysis\Codel\Images','',[120 170 220 270 10 20])
Trimming Images in folder \Users\Daniel\Documents\Deformation_Analysis\Codel\Images
```

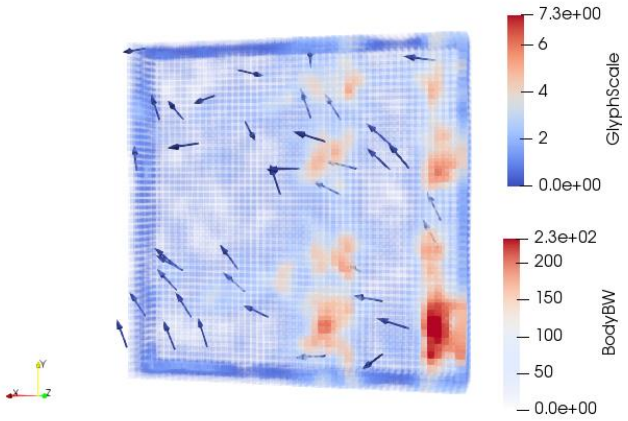



*Figura 21: Resultado 2 VTKPIV*

### 6.3.5.3. Resultado 3

Finalmente, el último Resultado que se obtiene es sobre la totalidad de la mosca frutera *Drosophila melanogaster*.

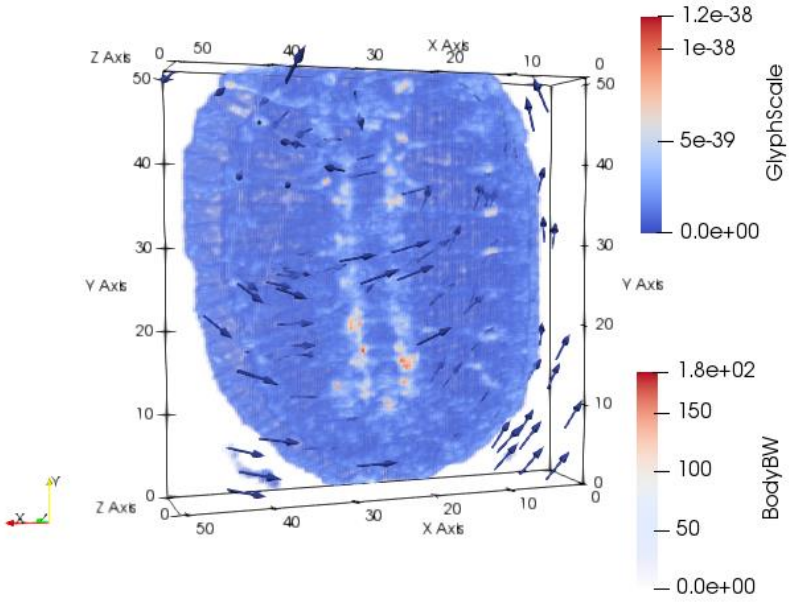



*Figura 22: Resultado 3 VTKPIV*

## 6.4. PIVError

*PIVError* grafica el error relativo del algoritmo. Se genera una imagen artificial 3D que se va desplazando en el tiempo a partir de un vector *u*. Cuando se obtienen las imágenes en una carpeta pasan por el algoritmo y se obtiene el vector desplazamiento. El error consiste en ver la diferencia existente entre ese vector desplazamiento (vector real) y el vector artificial.

Las funciones que se utilizan en *PIVError* son: *ADI* (Artificial Displacement Image) junto con *DisplacementImage*, que generan imágenes del tipo *Tif*, las características de éstas se especifican en *ParametersArtificialDisplacement*; y *PIVError* que calcula el error relativo.

La imagen que se crea en *ADI* tiene forma rectangular de dimensiones *xpixel y ypixel*, en el centro se dibuja un núcleo que se diferencia del resto de la imagen con un vector p. En la figura siguiente se muestra de forma esquemática la imagen [Figura 23], también, se van formando sobre el eje z, generando una imagen 3D con profundidad.

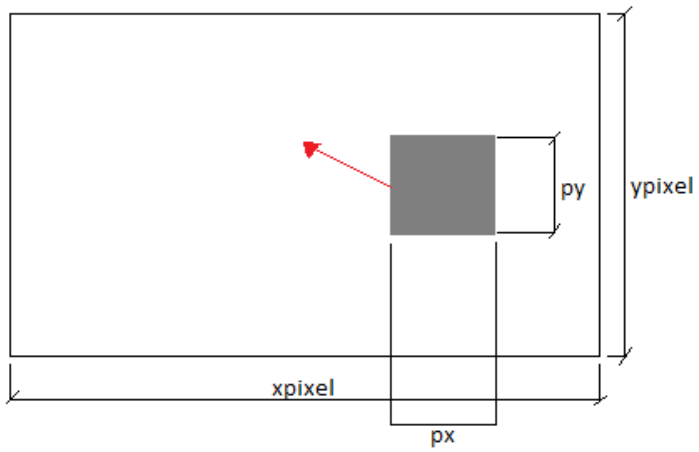
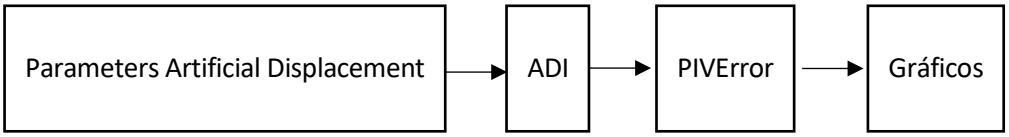



*Figura 23: Esquema en dos dimensiones de la imagen generada*

### 6.4.1. Función




Primer paso: Si *PIV_Error* está activado en *MainPIV*, entonces se procede al cálculo del error. En *ParametersArtificialDisplacemen*t, se introduce el tamaño y las dimensiones físicas reales de la imagen, el tiempo y el vector que genera el desplazamiento. También, con un vector *p* se delimita el espacio donde queda la marca central con una intensidad de gris distinto al resto.

Segundo paso: *ADI* genera los *Tifs* con el desplazamiento. Con un bucle en las dimensiones que se han introducido como parámetro, para el primer tiempo, se crea una matriz *image* dimensiones *xpixel* y *ypixel* por cada nivel de *z.* Sobre la imagen generada en el primer tiempo se aplica el desplazamiento en la "*marca*" con *DisplacementImage*.

Tercer paso: Aplicar el desplazamiento con *DisplacementImage,* se trata de que aquellos valores de la imagen cuya correlación es más grande a *0 (nivel negro)* aplicarles un desplazamiento *u* y una vez realizada esta operación guardar éstas en una carpeta.

Cuarto paso: Cálculo del error relativo sobre el desplazamiento que tienen las imágenes. Antes de entrar en materia acerca de *PIVError*, se debe dar una pincelada al concepto de error relativo y cómo éste se aplica sobre el programa. El error relativo presenta una forma matemática [Ec.24], la relación entre la suma de las diferencias entre el vector real $u_{real}$ y el vector artificial $u_{art}$ y, la multiplicación del número de nodos $nnodos$ y el vector artificial $u_{art}$.

$$\varepsilon_r = \frac{\sum u_{real} - u_{art}}{nnodos * u_{art}} \quad (24)$$

Las variables que se obtienen de entrada son: $u_{real}$ es el vector desplazamiento que se calcula en el algoritmo con tamaño *i filas* y *j columnas*; $u_{art}$ es el vector desplazamiento artificial y presenta las mismas dimensiones que $u_{real}$; $nnodos$ es el número total de nodos que hay en la imagen. Esta fórmula (fórmula error relativo) pasado a algoritmo, tiene un juego de operaciones entre matrices ya que las variables de entrada son vectores y, éstos funcionan como una matriz.

Los errores relativos de los desplazamientos conseguidos en FIDVC corresponden a cada una de las dimensiones del 3D, es decir, sobre el eje X, el eje Y y el eje Z existe un error relativo para cada uno de ellos, independientes el uno del otro.

Quinto paso: Graficar el error obtenido para cada tiempo el *subplot* de Matlab

### 6.4.2. Tabla de variables de entrada y salida en las funciones de VTKPIV

| **Función** | **Variables de entrada** | **Variables de salida** |
|---|---|---|
| **PIVError** | Uart; u; uMask; tmax; pSizes; L | EDisp |

*Tabla 4: Inputs & Outputs PIVError*

## 6.5. Error en Fast Iterative Digital Volume Correlation

El cálculo del error generado por FIDVC en el código PIV se obtiene gracias a la función *PIVError* explicada en el apartado anterior. Ahora, es el momento de obtener los resultados del error relativo en función de alguno de los parámetros escritos en ParametersArtificialDisplacement, este error se grafica en función del tiempo.

La configuración que toma el menú principal de MainPIV es la siguiente: Los parámetros utilizados son los ParametersArtificialDisplacement; El valor de PIV=2, que equivale a la técnica óptica FIDVC; Set.Inverse=true; y Las otras variables se mantienen nulas.

Se realizan diferentes pruebas para poder entender el comportamiento de FIDVC. La primera prueba consiste en cambiar el tamaño de la imagen que tiene el desplazamiento, manteniendo el vector velocidad constante. La segunda prueba consiste en cambiar el vector velocidad y mantener constante el tamaño de la imagen que tiene el desplazamiento. Cada prueba tiene tres experimentos, cada experimento tiene tres errores relativos sobre cada uno de los ejes del espacio, el eje X, el eje Y y el eje Z. El tamaño del Sub-Volumen de referencia se mantiene constante con un tamaño por defecto de *sx=32, sy=32 y sz=32.*

### 6.5.1. Prueba 1

Variables: Dimensiones de la imagen.

Constantes: Desplazamiento; Dimensiones de la marca; El tiempo.

Tiempo: 5 segundos

| | Dimensiones de la imagen | | | Desplazamiento | | | Dimensiones de la marca | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | xpixel | ypixel | zpixel | x | y | z | xmin | xmax | ymin | ymax | zmin | zmax |
| E1 | 30 | 30 | 30 | 5 | 5 | 0 | 5 | 10 | 5 | 10 | 5 | 10 |
| E2 | 50 | 50 | 50 | 5 | 5 | 0 | 5 | 10 | 5 | 10 | 5 | 10 |
| E3 | 40 | 40 | 40 | 5 | 5 | 0 | 5 | 10 | 5 | 10 | 5 | 10 |

*Tabla 5: Datos Prueba 1*

#### 6.5.1.1. Resultados

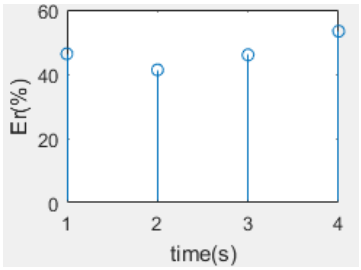
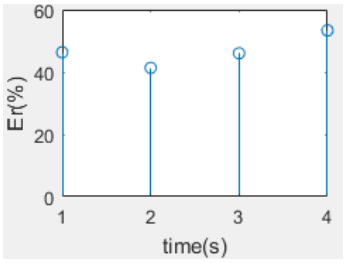
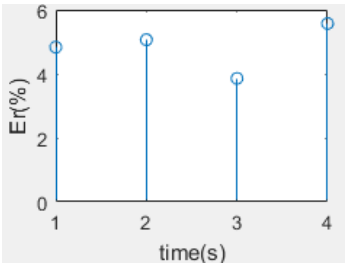
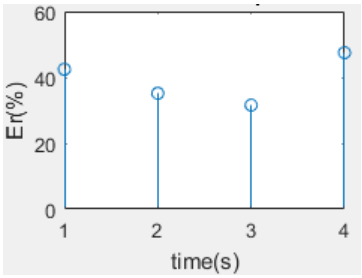
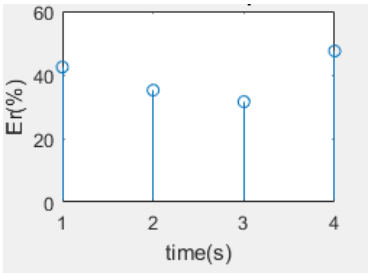
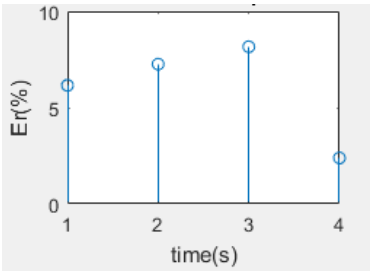
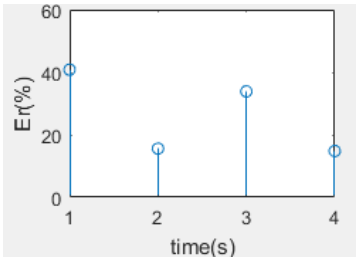
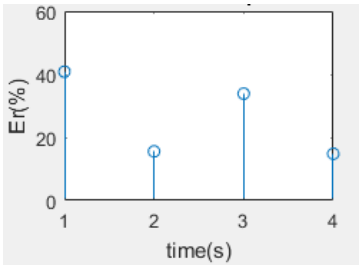
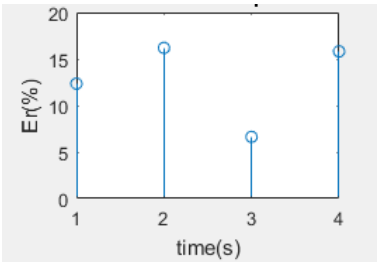
| | Error relativo sobre el Eje X | Error relativo sobre el Eje Y | Error relativo sobre el eje Z |
|---|---|---|---|
| E1 |  |  |  |
| E2 |  |  |  |
| E3 |  |  |  |

*Tabla 6: Resultados Prueba 1*

### 6.5.2. Prueba 2

Variables: Desplazamiento.

Constantes: Dimensiones de la imagen; Dimensiones de la marca; El tiempo.

Tiempo: 4 segundos

| | Dimensiones de la imagen | | | Desplazamiento | | | Dimensiones de la marca | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | xpixel | ypixel | zpixel | x | y | z | xmin | xmax | ymin | ymax | zmin | zmax |
| E1 | 50 | 50 | 50 | 5 | 5 | 0 | 5 | 10 | 5 | 10 | 5 | 10 |
| E2 | 50 | 50 | 50 | 10 | 10 | 0 | 5 | 10 | 5 | 10 | 5 | 10 |
| E3 | 50 | 50 | 50 | 12 | 12 | 0 | 5 | 10 | 5 | 10 | 5 | 10 |

*Tabla 7: Datos Prueba 2*

#### 6.5.2.1. Resultados

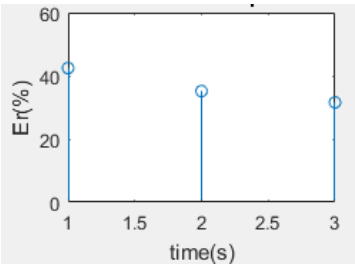
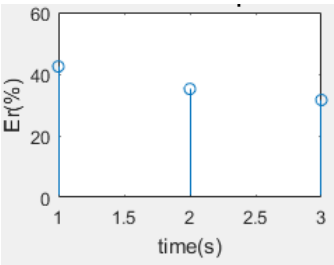
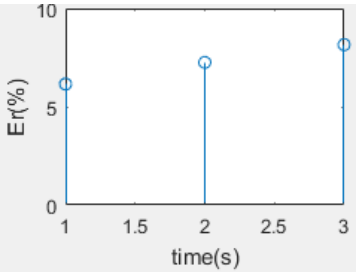
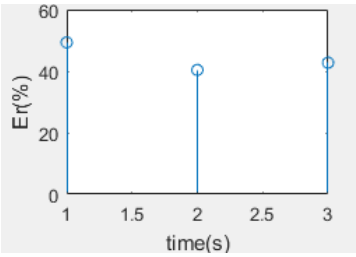
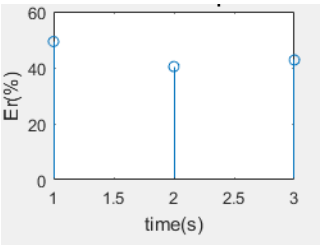
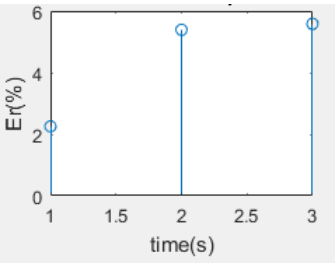
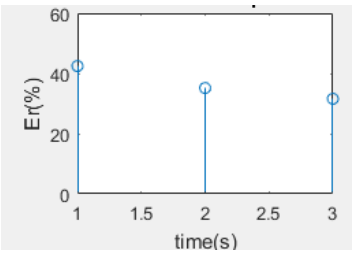
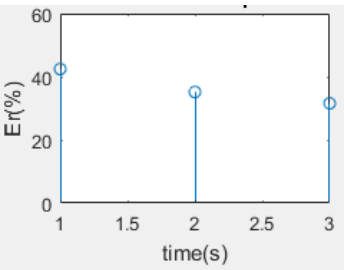
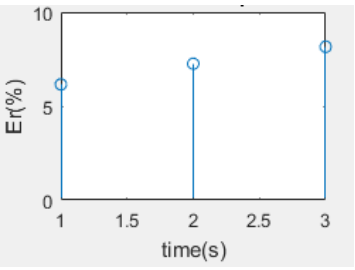
| | Error relativo sobre el Eje X | Error relativo sobre el Eje Y | Error relativo sobre el eje Z |
|---|---|---|---|
| E1 |  |  |  |
| E2 |  |  |  |
| E3 |  |  |  |

*Tabla 8: Resultados Prueba 2*

# Análisis impacto ambiental

El TFG es un trabajo realizado con ordenador, básicamente los algoritmos generados no presentan ningún tipo de impacto ambiental.

# Conclusiones

Una primera conclusión es referente al programa de Post-Procesado Paraview. Podemos decir que tiene una gran capacidad para la representación gráfica, presentando un formato aparentemente sencillo, pero que llega a tener un gran número de herramientas para mejorar la calidad del Post-Procesado que se quiera realizar. La idea de VTK es muy intuitiva en el momento de organizar los datos. Poder formar una malla con diferentes figuras geométricas permite construir cuerpos complejos.

La representación del desplazamiento en formato VTK se ha realizado correctamente, pero no se ha visto del todo si el vector representado en Paraview presenta la magnitud del desplazamiento correcta, sí que indica la dirección y el sentido, y cuando no se aplica el desplazamiento sobre los vectores, sino que, sobre el volumen o la superficie, se observa la magnitud del desplazamiento correcta (el volumen o superficie son herramientas de Paraview para visualizar los resultados)

Se han tenido dificultades en el momento de representar las tracciones en los cuerpos, puede que sea un error de formato de VTK o que el cálculo no se realiza del todo bien, pero éstas se mantenían constantes a lo largo del tiempo. Esta ha sido la razón por la que las tracciones no se hayan visto en ninguno de los apartados.

Los errores relativos sobre los ejes en FIDVC son muy elevados, ésto nos da a entender que puede haber algún tipo de problema en el filtrado de la imagen o de los datos del desplazamiento. Igualmente, en estas pruebas no se han tocado las tolerancias, ya que era la creación de una imagen bastante limpia. También, vemos que el error relativo obtenido puede que sea elevado debido al tamaño del sub-volumen que por defecto es de *32x32x32*, si la marca que se desplaza es muy pequeña o muy grande respecto de este sub-volumen puede generar errores. El código PIV no permite de forma directa cambiar estas dimensiones, sino que lo hace automáticamente en función de las imágenes que se estudian.

Por un lado, en los resultados, cuando el tamaño de la imagen va aumentando gradualmente no produce unos cambios importantes en los valores del error relativo, más o menos el error se mantiene estable. Sin embargo, en los resultados, cuando el vector velocidad va aumentando se produce un aumento del error relativo en el desplazamiento, esto es porque el sub-volumen encargado de la correlación no se desplaza a la misma velocidad que el desplazamiento introducido.

# Presupuesto

En este apartado del trabajo se expone el presupuesto. A continuación, se verán una serie de tablas en donde se muestra detalladamente todo el presupuesto, el cual se divide en distintas partidas, incluyendo Mano de Obra y Otros gastos.

**Mano de Obra**

En la mano de obra se tienen en cuenta las horas trabajadas a lo largo del proyecto. Cada semana se han empleado 10 horas de trabajo. En septiembre del 2017 se inició este proyecto y acabó en el mes de mayo del 2018, por lo que ha requerido un total de 32 semanas, esto significa que se han realizado un total de 320 horas en la comprensión y desarrollo del trabajo, lectura de artículos, lectura y pruebas en los algoritmos, desarrollo de las funciones del algoritmo y de los documentos de trabajo.

Para el cálculo del presupuesto se ha tomado como referencia el salario medio base de un ingeniero junior (menos de dos años de experiencia) en España, esto es un sueldo neto de 40.000€ al año, una cantidad que equivale a 21€ la hora. En la siguiente tabla se muestra la cantidad de horas trabajadas en las actividades realizadas y, el precio total por cada una de éstas. Finalmente, la suma de todos los precios en las actividades es el precio total presupuestado para el proyecto.

| Trabajo / Tareas | Horas | Euros / hora | Precio en Euros |
|---|---|---|---|
| Lectura artículos | 40 | 21 € | 840 € |
| Lectura algoritmos | 80 | 21 € | 1.680 € |
| Pruebas algoritmo | 50 | 21 € | 1.050 € |
| Desarrollo de las funciones | 120 | 21 € | 2.520 € |
| Documentos trabajo | 30 | 21 € | 630 € |
| Total | 320 | | 6.720 € |

*Tabla 9: Presupuesto*

Como se puede observar, la tarea que supone un mayor coste económico es el Desarrollo de las funciones, concretamente 2.520€ ya que es la función que requiere un mayor número de horas de dedicación por parte del ingeniero. En segundo lugar, encontramos la Lectura de algoritmos, que supondrá un coste de 1.680€

Por el contrario, las tareas relativas a los Documentos de trabajo son las que requiere un menor número de horas y en consecuencia es la partida con menor coste del presupuesto, concretamente 630€.

# Bibliografía

1. *Drosophila Melanogaster.* s.l. : Wikipedia.

2. *Inverse Mechanical Analysis of Brain Growth in Embryo Development.*

3. Lucas, Bruce D. y Kanade, Takeo. *An Iterative Image Registration Technique with an Application to Stereo Vision.* Computer Science Department Carnegie Melión University . Pittsburgh, Pennssylvania : s.n., April 1981.

4. Westerweel, J. y Poelma, C. *Introduction of Paticle Image Velocimetry.* School of Engineering. Maryland, College Park : s.n.

5. Pan, Bing, Dafang, Wu y Zhaoyang, Wang. *Internal displacement and strain measurement using digital volume correlation.* Department of Mechanical Engineering. Washington, DC : s.n., 2012 October.

6. ·C.Franck, E. Bar-Kochba · J. Toyjanova · E. Andrews · K.-S. Kim. *A Fast Iterative Digital Volume Correlation Algorithm for Large Deformations.* Experimental Mechanics. August 2014. DOI 10.1007/s11340-014-9874-2.

7. Prasad, Ajay K. *Particle image velocimetry.* Department of Mechanical Engineering, University of Delaware, Newark, DE 19716-3140, USA. July 2000.

# Anexo A

Códigos creados para Matlab.

## Tifs2VTK

```
function Tifs2VTK(name,Lx,Ly,Lz,tmax,xpixel,ypixel,zpixel)
% File for generating VTK files *.vtk from tiff images for
% INPUT:
% name = folder wher images z1t1.tif ... zZtT.tif are located.
%         Z=number of z-layers, T=number of time instants.
% xpixel,ypixel,zpixel=size image
% Lx,Ly,Lz: size of whole domain in microns
% tmin,max=initial and final
zmax=zpixel;
nx=xpixel;
ny=ypixel;
nz=zpixel;
dx=Lx/nx;
dy=Ly/ny;
dz=Lz/nz;
D=[dx dy dz];
nx=nx-1;  %division's domain
ny=ny-1;  %division's domain
nz=nz-1;
%Gebenerate a Mesh with input values
[C,X]=GenerateMesh(D(1),D(2),D(3),nx,ny,nz); % Dimension [dx*(nx-1),dy*(ny-
1),dz*(nz-1)]
ncells=size(C,1);
npoints=size(X,1);
GrayLevel=zeros(npoints,1);
nnodes=size(C,2);
%Read images and determination Gray Level
for t=1:tmax
    for z=1:zmax
        FileName=strcat(name,Esc,'z',num2str(z),'t',num2str(t),'.tif');
        image=double(imread(FileName));
        GrayLevel=AssignGrayLevel(image,GrayLevel,z,nx,ny);
    end
    % WRITE VTK FILE
     fid=fopen(strcat('Imagen','t',num2str(t),'.vtk'),'w');
     fprintf(fid,'# vtk DataFile Version 3.98\n');
     fprintf(fid,'Image_Correlation\n');
     fprintf(fid,'ASCII\n');
     fprintf(fid,'DATASET UNSTRUCTURED_GRID\n');
    % Write nodal coordinates
     fprintf(fid,'POINTS %i float\n',npoints);
     for n=1:npoints
         fprintf(fid,'%e %e %e \n',X(n,:));
     end
    % Write elemental connectivity
```

```
    fprintf(fid,'CELLS %i %i\n',ncells,ncells*nnodes);
    for e=1:ncells
         fprintf(fid,'%i ',nnodes-1,C(e,1:8)-1);
         fprintf(fid,'\n');
    end
    fprintf(fid,'CELL_TYPES %i\n',ncells);
    for e=1:ncells
        fprintf(fid,'%i\n',12);
    end
    % Write GrayLevel
    fprintf(fid,'POINT_DATA %i \n',npoints);
    fprintf(fid,'SCALARS BodyBW float \n');
    fprintf(fid,'LOOKUP_TABLE default \n');
    for n=1:npoints
        fprintf(fid,'%i\n',GrayLevel(n,:));
    end
    % Close file
     fclose(fid);
end
```

## AssignGrayLevel

```
function GrayLevel=AssignGrayLevel(image,GrayLevel,z,nx,ny)
% ASSIGN VALUE OF IMAGE TO GRAYLEVEL'
%USAGE:
%AssignGrayLevel(image,GrayLevel,z,nx,ny);
%INPUTS:
%image= 2D Image in a certain time and layer
%GrayLevel= Zero Matrix (nnodes,1)
%z= Number of layers
%nx,ny= vertical and horizontal domain's divisions
%OUTPUTS:
%GrayLevel=Level of gray for each pixel and node
 for j=1:ny+1
     for i=1:nx+1
         d=(z-1)*(ny+1)*(nx+1)+(j-1)*(nx+1)+i;  %Nodos
         GrayLevel(d,1)=image(i,j);
     end
 end
end
```

## VTKPIV

```
function
uMaskL=VTKPIV(name,C,c,cMask,T,tMask,tmin,tmax,u,uMask,u0Mask,u0MaskL,x,X,Lagr
angian)
fprintf('Writing VTK results\n');
if tmax>=tmin
    for t=tmin:tmax
```

```
        VTKFileS=strcat(name,'S',sprintf('%i',t),'.vtk');
        VTKFileSJ=strcat(name,'SJ',sprintf('%i',t),'.vtk');
        VTKFileL=strcat(name,'L',sprintf('%i',t),'.vtk');
        VTKFileLJ=strcat(name,'LJ',sprintf('%i',t),'.vtk');
        nnodes=size(X,1);
        nt=tmax-tmin+1;
        tic;
        % fid=0;
        %
fid=VTKRes(fid,nameVTK,true,0,[],zeros(nodes,1),[],[],[],zeros(nodes,3),[],[],
[]);
        uMaskL=ones(nnodes,1);
        VTKfidS=fopen(VTKFileS,'w');
        VTKfidSJ=fopen(VTKFileSJ,'w');
        VTKMesh(C,X,VTKfidS);
        VTKMesh(C,X,VTKfidSJ);
        if Lagrangian % Find nodes that in final domain ar inside initial
image domain
            uMaskL=FindMaterialDomain(C,X,x(:,:,tmax));
            VTKfidL=fopen(VTKFileL,'w');
            VTKfidLJ=fopen(VTKFileLJ,'w');
            VTKMesh(C,X,VTKfidL,umaskL);
            VTKMesh(C,X,VTKfidLJ,umaskL);
        end
        fprintf('t=%i / %i, elapsed time=%f \n',t-tmin+1,nt,t);

VTKRes(VTKfidS,false,t,X,Grid2Mat(c(:,:,:,t)),cMask(:,t),T(:,:,t),tMask(:,t),u
(:,:,t),uMask(:,t),u(:,:,t),u0Mask(:,t));

%VTKRes(VTKfidSJ,false,t,X,Grid2Mat(c(:,:,:,t)),cMask(:,t),T(:,:,t),tMask(:,t)
,u(:,:,t),uMask(:,t),u(:,:,t),u0Mask(:,t));
        if Lagrangian

VTKRes(VTKfidL,false,t,X,Grid2Mat(c(:,:,:,t)),cMask(:,t),[],[],x(:,:,t)-
X,uMaskL,x(:,:,t)-X,u0MaskL(:,t),Lagrangian);

VTKRes(VTKfidLJ,false,t,X,Grid2Mat(c(:,:,:,t)),cMask(:,t),[],[],x(:,:,t)-
X,uMaskL,x(:,:,t)-X,u0MaskL(:,t),Lagrangian);
        end
        fclose(VTKfidS);
        %fclose(VTKfidSJ);
        if Lagrangian
            fclose(VTKfidL);
            fclose(VTKfidLJ);
        end
    end
end
```

## VTKRes

```
function VTKfid=VTKRes(VTKfid,New,time,X,c,cMask,t,tMask,u,uMask,u0,u0Mask,~)
%MATLAB2GID Summary of this function goes here
```

```
% INPUT:
% u=nodal displacemetns in matrix form:
% u=[ux_1 uy_1 uz_1
%    ux_2 uy_2 uz_2
%    ...
%    ux_N uy_N uz_N]
%
% t= traction values, same format as u.
% uMask(i)=0 displacements and correlation of node i not written
% MESH
Lagrangian=nargin>13;
if Lagrangian
    displ='DispLagrangian';
    displApp='AppliedDispLagrangian';
    nodes0=size(u,1);
else
    displ='Displacements';
    displApp='ApplDisplacements';
    nodes0=0;
end
if ~exist('cMask','var') || min(size(cMask))==0
    cMask=ones(size(c,1),1);
end
if ~exist('uMask','var') || min(size(uMask))==0
    uMask=ones(max([size(u,1),size(X,1),size(u0,1)]),1);
end
if ~exist('u0Mask','var') || min(size(u0Mask))==0
    u0Mask=ones(size(u0,1),1);
end
if ~exist('tMask','var') || min(size(tMask))==0
    tMask=ones(size(t,1),1);
end
% Displacements
nnodes=size(u,1);
if nnodes>0
    fprintf(VTKfid,'POINT_DATA %i\n',nnodes);
    fprintf(VTKfid,'VECTORS  %s float\n',displ);
    for i=1:nnodes
        if uMask(i)>0
            fprintf(VTKfid,'%e %e %e',u(i,:));
            fprintf(VTKfid,'\n');
        end
    end
end
% Applied Displacement
nnodes=size(u0,1);
if nnodes>0
   % fprintf(fid,'POINT_DATA %i\n',nnodes);
   if u0Mask(i)>0 && (uMask(i)>0 || ~Lagrangian)
      fprintf(VTKfid,'VECTORS  %s float\n',displ);
       for i=1:nnodes
           fprintf(VTKfid,'%e %e %e',u0(i,:));
           fprintf(VTKfid,'\n');
       end
```

```
   end
end
% Positions for visualising deformation
if exist('X','var')
    nnodes=min(size(X,1),size(u,1));
    if nnodes==0
        nnodes=min(size(X,1),size(c,1));
    end
    % Displacements
    PosL='PositionsL';
    Pos='Positions';
    if nnodes>0
        if Lagrangian
            fprintf(VTKfid,'VECTORS  %s float\n',PosL);
        else
            fprintf(VTKfid,'VECTORS  %s float\n',Pos);
        end
        for i=1:nnodes
            if size(u,1)>0
                tol=norm(u(i,:));
            elseif size(c,1)>0
                tol=norm(c(i,:));
            end
            if tol>eps && (uMask(i)>0 || ~Lagrangian)
                fprintf(VTKfid,'%e %e %e',X(i,1),X(i,2),X(i,3));
                fprintf(VTKfid,'\n');
            else
                fprintf(VTKfid,'%e %e %e',0,0,0);
                fprintf(VTKfid,'\n');
            end
        end
    end
end
% Tractions
Trct='Traction';
nnodes=size(t,1);
if nnodes>0
    fprintf(VTKfid,'VECTORS %s float\n',Trct);
    for i=1:nnodes
        if tMask(i)>0
            fprintf(VTKfid,'%e %e %e',t(i,:));
            fprintf(VTKfid,'\n');
        else
            fprintf(VTKfid,'%e %e %e',0,0,0);
            fprintf(VTKfid,'\n');
        end
    end
end
% Correlation
CorrL='CorrelationsL';
Corr='Correlation';
if nnodes>0
    if ~exist('uMask','var')
        uMask=ones(nnodes,1);
```

```
    elseif min(size(uMask))==0
        uMask=ones(nnodes,1);
    end
    if Lagrangian
        fprintf(VTKfid,'SCALARS %s float 1\n',CorrL);
    else
        fprintf(VTKfid,'SCALARS %s float 1\n',Corr);
    end
        fprintf(VTKfid,'LOOKUP_TABLE default\n');
    for n=1:nnodes
        if c(i)>0
        fprintf(VTKfid,'%e \n',c(i));
        else
        fprintf(VTKfid,'%e \n',0);
        end
    end
end
end
```

## VTKMesh

```
function VTKMesh(C,X,fid,xMask,tmax);
% Write in VTKFile the mesh part of the results
% xMask(i)=1: node i should be written in Lagrangian mesh
% xMask(i)=0: node i should not be written in Lagrangian mesh, and neither
%             any element including this node.
% u=[node direction value] all nodal displacements
% t=[node direction value] traction values for computed nodes
[npoints,dim]=size(X);
[nelem,nnod]=size(C);
nnod=nnod-1;
% MESH
Lagrangian=nargin>4;
if dim==3
    nnodes=size(C,2);
    type=12;
    ncells=size(C,1);
else
    nnodes=1;
    T=(1:npoints);
    type=1;
    ncells=npoints;
end
% Write coordinates
fprintf(fid,'# vtk DataFile Version 3.98\n');
fprintf(fid,'Vertex_vtk\n');
fprintf(fid,'ASCII\n');
fprintf(fid,'DATASET UNSTRUCTURED_GRID\n');
if Lagrangian && max(size(xMask))==0
    xMask=ones(size(X,1),1);
end
```

```
if Lagrangian
    nnodes=size(X,1);
    nnode=size(C,2)-1;
    nele=size(C,1);
    fprintf(fid,'POINTS %i double\n',nnodes);
    for i=1:nnodes
        if xMask(i)>0
            fprintf(fid,'%e %e %e\n ',X(i,:));
        else
            fprintf(fid,'%e %e %e\n',NaN);
        end
    end
    eMask=zeros(nnodes,1); % Mask for LAgranagian particles
    MatP=2*ones(nnodes,1);
    fprintf(fid,'CELLS %i %i\n',ncells,ncells*(nnode-1));
    for i=1:nelem
        if sum(xMask(C(i,:),:))==nnode
            fprintf(fid,'%i ',nele+i,C(i,1:end-1)+nnodes,C(i,end)+3);
            fprintf(fid,'\n');
            eMask(C(i,1:end-1))=1;
            if C(i,end)==1
                MatP(C(i,1:end-1))=1; % Mark oarticles as principal (harder)
material
            end
        end
    end
    fprintf(fid,'CELL_TYPES %i\n',ncells);
    for e=1:ncells
       fprintf(fid,'%i\n',type);
    end

% Write MESH for normal values
else
    fprintf(fid,'POINTS %i double\n',npoints);
    for n=1:npoints
       fprintf(fid,'%e %e %e \n',X(n,:));
    end
    % Write connectivity
    fprintf(fid,'CELLS %i %i\n',ncells,ncells*nnodes);
    for e=1:ncells
       fprintf(fid,'%i ',nnodes-1,C(e,1:8)-1);
       fprintf(fid,'\n');
    end
   % Write type of element
    fprintf(fid,'CELL_TYPES %i\n',ncells);
    for e=1:ncells
       fprintf(fid,'%i\n',type);
    end
end
 end
```

## ADI

```
function ADI(xpixel,ypixel,zpixel,Lx,Ly,Lz,p,u,time)
%INPUTS:
%time=Max. Time for the experimental system
%zmax=Max. Layer in the experimental system
%xpixel= horizontal pixels image
%ypixel= vertical pixels image
%p=stain's size p[xint xfin yint yfin zint zfin]
%u1 X Displacement
%u2 Y Displacement
%u3 Z Displacement
%OUTPUTS:
%Array ArtificialImages6 with images
%'z1t1.tif'
%   ....
%   ....
%'zntn.tif'
t=1;
for z=1:zpixel
      image=zeros([xpixel,ypixel],'uint8');
         if p(5)<=z & z<=p(6)
           for y=p(3):p(4);
              for x=p(1):p(2);
                  image(x,y)=120;
              end
            end
        end
filename=strcat('z',num2str(z),'t',num2str(t),'.tif');
imwrite(image,fullfile('c:\','Users\','Daniel\','Documents\','Deformation_Anal
ysis\','Code3_VTK\','ArtificialExperimental',filename));
end
% Image Displacement
for t=2:time

name='C:\Users\Daniel\Documents\Deformation_Analysis\Code3_VTK\ArtificialExper
imental';
    DisplacementImage(name,t,zpixel,u);
end
end
```

## DisplacementImage

```
function DisplacementImage(name,t,zmax,u)
%DISPLACEMENTIMAGE creats a displacement in the stain created
% INPUT
%  name = Folder with tifs
%  t = Time
%  zmax = Number of layers
%  u = Displacement vector,in three directions
%
```

```
% OUTPUT
%  Folder with all the tifs created with a displacement in the stain  created
%
% NOTE: For the moment is not created the z dispplacement
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for z=1:zmax
   t=t-1;
   FileName=strcat(name,Esc,'z',num2str(z),'t',num2str(t),'.tif');
   image=double(imread(FileName));
   [xmax,ymax]=size(image);
   imaged=zeros([xmax,ymax],'uint8');
   for y=1:ymax
       for x=1:xmax
           if image(x,y)>0
              imaged(x+u(1),y+u(2))=image(x,y);
           end
       end
   end
   t=t+1;
   filename=strcat('z',num2str(z),'t',num2str(t),'.tif');

imwrite(imaged,fullfile('c:\','Users\','Daniel\','Documents\','Deformation_Ana
lysis\','Code3_VTK\','ArtificialExperimental',filename));
end
```

## PIVError

```
function [ErDispl]=PIV_Error(u0,u,uMask,tmax,pSizes,L)
% Function errors in FIDVC
% INPUT:
%  u0: Artificial displacement
%  u: Displacement FIDVC
%  uMask: Filter Displacement
%  pSizes:xpixel, ypixel, zpixel
%  L: Real domain's dimension
%  tmax: Time
% OUTPUT:
%  Plots with Errors: X error, Y error, Global Error
%%%%%%
u0=u0.*[L(1)/pSizes(1) L(2)/pSizes(2) L(3)/pSizes(3)];
ErDispl=zeros(tmax-1,size(u,2));
nnodes=size(u(:,:,tmax),1);
ua=zeros(nnodes,size(u(:,:,tmax),2));
  for i=1:nnodes
      ua(i,1)=u0(1);
      ua(i,2)=u0(2);
      ua(i,3)=u0(3);
  end
for t=1:tmax-1
Ea(:,:,t)=abs(u(:,:,t)-ua);
ErDispl(t,1)=(sum(Ea(:,1,t))/(nnodes*ua(1)))*100; % Relative Error X
ErDispl(t,2)=(sum(Ea(:,2,t))/(nnodes*ua(2)))*100; % Relative Error Y
```

```
ErDispl(t,3)=(sum(Ea(:,3,t))/(nnodes*ua(3)))*100; % Relative Error Z
end

end
```

## Main_PIV

```
% Script for retrieving displacement field from 3D correlation and
% generation of output for GID post-processing.
%
% Script assumes a folder with name 'nameTifs' with a set of tif images that
% contains the z planes. Each tif file is named as:
%
% z1t1.tif   : first  cross-section at time t1
% z2t1.tif   : second cross-section at time t1
% ...
% z1t2.tif   : first cross-section at time t2
% ...
% Jose Munoz, 2016
%
% COMMENTS:
% Tolerances and Masks (c=gray level, cmax=max(c) of first image.):
%
% TolC: Prints Images with c>TolC*max(c) for each time-step
%       Creates u0Mask (list of nodes with applied displ) ic c>TolC*maxc
% TolU: in xcorr3 (not FIDVC when PIV=2) sets c=0 if c(I)<tolU*cmax or
c(J)<tolU*cmax
% TolM: creates mMask, which determines soft material for c(i)<TolM*cmax
% tolE: uMask(i)=0 if node i has c<tolE*cmax. Only set if Set.FilterDisp=true
%       eMask(e)=0 if all nodes of element c<tolE*cmax. Element not
%                  included in inverse. Only set when Set.FilterDisp=true
%
clearvars
SetPaths();
% Name of lsm file and size in pixels
%ParametersSingapore();
%ParametersElena();
%ParametersArtificial1D();
% ParametersArtificialExp();
% ParametersImage8();
% ParametersMYRRot();
ParametersFasII();
% ParametersArtificialDisplacement();
% ParametersMYR();
% ParametersRun4();
% ParametersPicamal();
PIV_Error=false; %Relative Error in PIV, this option must be with
ParametersArtificialDisplacement
WriteGIDImages=false; % true=Write files for visualising whole images in GID
WriteVTKImages=false;% true=Write files for visualising whole images in
Paraview
RotateImages=false;   % Remove rigid body rotations
```

```
WriteRotatedImages=false;   % Write Rotated tif Images. If RotateImages==1 &&
WriteRotatedImages=0, images are rotated in PIV_CNS, and not written.
PIV=2;              % 0=Donot compute displacements
                        % 1=Compute 3D PIV (Eulerian displacements) using xcorr
                        % 2=Compute 3D PIV using FIDVC
Set.Inverse=true;     % Use inverse analysis to complete displacements on
domain
Set.Lagrangian=false;  % Compute and visualise tractions on deformed shape
%Set.FilterDisp=false; % Compute  displacements for only those elements that
have a correlation > TolE*max(correlation)
%Set.FilterImages=[2 2 1];  % Image is low resoluted by averaging values in
boxes of size (in pixels) in Set.FilterImages
                            % If Set.Images(i)==1, no filtering in direction
i
Set.Plot=false; %Plots displacements, deformations and tractions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Initialise VARIABLES and call relevant functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SetDefaultsPIV();
% TrimImages(name,folderout,[9 20 40 54 12 16]);
% Apply filtering
if PIV_Error
    fprintf('Doing the Artificial Image Displacement \n');
    ADI(xpixel,ypixel,zpixel,Lx,Ly,Lz,p,uart,time);

name='C:\Users\Daniel\Documents\Deformation_Analysis\Code3_VTK\ArtificialExper
imental';
end
SetDefaultsPIV();
if max(Set.FilterImages)>1
    folderIn=name;
    name=strcat(folderIn,'Filter');

[xpixel,ypixel,zpixel]=Filter(folderIn,name,Set.FilterImages,Set.FilterReduce)
;
end
%
if Set.WriteRotatedImages && Set.RotateImages
    folderIn=name;
    name=strcat(folderIn,'Rot');
    RotateWrite(folderIn,name,tmax);
end
% Write Images for GID
if WriteGIDImages
    if ~exist('xMin','var')
        xMin=1;
    end
    if ~exist('xMax','var')
        xMax=xpixel;
    end
    if ~exist('yMin','var')
        yMin=1;
    end
    if ~exist('yMax','var')
```

```
        yMax=ypixel;
    end
    if ~exist('zMin','var')
        zMin=1;
    end
    if ~exist('zMax','var')
        zMax=zpixel;
    end
    trim=[xMin xMax yMin yMax zMin zMax];

Tifs2GID(name,nametifs,Lx,Ly,Lz,tmin,tmax,xpixel,ypixel,zpixel,trim,Set.TolC);
end
if WriteVTKImages
   Tifs2VTK(name,Lx,Ly,Lz,tmax,xpixel,ypixel,zpixel);
end
% PIV analysis
% BEST SETTINGS so far with experimental data and applied displacement of 10
pixels:
% Uz=4
%  sx=sy=10, sz=5; si must be same order of displacement. N=0 in xcorr3
% Ux=Uy=10
%  sx=sy=15, sz=10; si must be same order of displacement.N>0 in xcorr3
%  Set.TolC=Set.TolU=0.5, Set.Inverse=1, Set.FilterDisp=0.
%  The lower values of TolC, the more displacemetn data is achieved
%  TolC give more
% [Ux Uy Uz] from experimental first 2 values
%  FilterImages=[4 4 2];
%  sx=sy=15, sz=10; si must be same order of displacement.N>0 in xcorr3
%  Set.TolC=0.4,Set.TolU=0.5, Set.Inverse=1, Set.FilterDisp=0.
% Create displacement field from 3D correlation (this may take long, 17
% hours in 1024x1024x31 pixel domain with 31 time-steps on Mac 2015)
if PIV
    % Overwrite default values
    Set.TolC=0.2; % Def = 0.1. For analysing images (unused)
    Set.TolU=0.1; % Def=0.1
    Set.TolE=0.1; % Def=0.1
    Set.TolM=0.2; % Def=0.2. Material 1 when c>TolM*cmax
    Opt=struct();
    % Overlapping
    if exist('ov','var'), Opt.ov=ov;end
    % Size of subdomains
    if exist('sx','var'), Opt.sx=sx;end
    if exist('sy','var'), Opt.sy=sy;end
    if exist('sz','var'), Opt.sz=sz;end
    L=[Lx Ly Lz];
    pSizes=[xpixel ypixel zpixel];
    Set.PIV=PIV;
    if PIV_Error % Cálculo del error generado en PIV
    PIV_CNS(name,nametifs,L,Opt,pSizes,tmin,tmax,Set,uart);
    else
    uart=[0 0 0];
    PIV_CNS(name,nametifs,L,Opt,pSizes,tmin,tmax,Set,uart);
    end
    if Set.Plot
```

```
        PlotsPIV(strcat(name,'.mat'))
    end
end
```

## PIV_CNS

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('Processing PIV\n')
Lx=L(1);
Ly=L(2);
Lz=L(3);
[sSizes,ov]=SetSizes(pSizes,Opt,Set);
xpixel=pSizes(1);
ypixel=pSizes(2);
zpixel=pSizes(3);
sx=sSizes(1);
sy=sSizes(2);
sz=sSizes(3);
% Read images
esc=Esc();
if isempty(strfind(name,esc))
    folder=pwd;
    if isempty(nametifs)
        folder=strcat(folder,esc,name,'Tifs',esc);
    else
        folder=strcat(folder,escc,name,nametifs,esc);
    end
else
    if isempty(nametifs)
        folder=strcat(name,'Tifs',esc);
    else
        folder=strcat(name,nametifs,esc);
    end
end
% Check that tif folder exist
if ~exist(folder,'dir')
    folder=strcat(folder(1:end-5),esc); % Remove 'Tifs' at end of string
end
Set.Lagrangian=Set.Lagrangian && Set.Inverse;
% Number of displ nodes along each direction x,y,z
image2=ReadImages(folder,tmin,zpixel); % Read first image
[D,nx,ny,nz,oSizes]=SetnSizes(Set.dm,image2,L,pSizes,sSizes,Set.PIV);
nt=tmax-tmin+1;
nSizes=[nx,ny,nz];
% Output to GID
[C,X]=GenerateMesh(D(1),D(2),D(3),nx-1,ny-1,nz-1); % Dimension [dx*(nx-
1),dy*(ny-1),dz*(nz-1)]
Set.dim=size(X,2);
nnodes=nx*ny*nz;
Set.nodes=nnodes;
eMask=[];
tMask =ones(Set.nodes,nt);
```

```
uMask =ones(Set.nodes,nt);
u0Mask=zeros(Set.nodes,nt);
cMask =zeros(Set.nodes,nt);
if nx<=1 || ny<=1 || nz<=1
    Set.Inverse=0;
end
% Initialise displacemens
ux=zeros(nx,ny,nz,nt);
uy=zeros(nx,ny,nz,nt);
uz=zeros(nx,ny,nz,nt);
c=zeros(nx,ny,nz,nt); % correlation
u=zeros(Set.nodes,Set.dim,nt); % spatial displacements
if Set.Inverse
    Set.d=D(1); % dx;
    Set.h=D(3)*nz;
    Set.E=1000;
    Set.v=0.3;
    Set.fid=0;
    if Set.Lagrangian
        u0MaskL=zeros(Set.nodes,nt);
    else
        u0MaskL=[];
    end
else
    u0MaskL=[];
end
tic;
if ~Set.WriteRotatedImages && Set.RotateImages
    image2=RotateImages(image2);
end
Set.TolU=Set.TolU*max(max(max(image2))); % Compute maximum pixel level. Taken
from 1st time step
x=zeros(size(X,1),size(X,2),tmax);
tM=x;
x(:,:,tmin)=X; % Material coord=Spatial coord
for t=tmin:tmax-1 % loop on time-steps
    if t==1
        cmax=max(max(max(image2)));
    end
    image1=image2;
    image2=ReadImages(folder,t+1,zpixel);
    if ~Set.WriteRotatedImages && Set.RotateImages
        image2=RotateImages(image2);
    end
    if Set.PIV==1  %xcorr PIV

[ux(:,:,:,t),uy(:,:,:,t),uz(:,:,:,t),c(:,:,:,t),cI]=funXcorr3(image1,image2,nS
izes,oSizes,pSizes,sSizes,Set,t,tmax);
    elseif Set.PIV==2 %
      [ux(:,:,:,t),uy(:,:,:,t),uz(:,:,:,t),c(:,:,:,t),cI] =
funIDVC2(Set.dm,image1,image2,sSizes, t,tmax);
    end
    ux(:,:,:,t)=ux(:,:,:,t)*Lx/xpixel;
    uy(:,:,:,t)=uy(:,:,:,t)*Ly/ypixel;
```

```
    uz(:,:,:,t)=uz(:,:,:,t)*Lz/zpixel;
    u(:,:,t)=Grid2Mat(ux(:,:,:,t),uy(:,:,:,t),uz(:,:,:,t)); % displacements in
um, not element size

[cMask(:,t),mMask,u0Mask(:,t)]=MaskCU(Grid2Mat(c(:,:,:,t)),Grid2Mat(cI),cmax,u
(:,:,t),Set.TolC,Set.MTol);
    if Set.FilterDisp % Filter elements that have all nodes with very low
correlation
        [eMask,uMask(:,t)]=MaskE(cmax,Grid2Mat(cI),C,X,Set.TolE); % Filter
elements with nodes that have low correlation
    end
    fprintf('Obtained %i dof of a total of
%i.\n',sum(u0Mask(:,t)>0)*Set.dim,Set.dim*sum(uMask(:,t)>0));
    % Compute Mechanical Inverse
    if Set.Inverse
        uD=[]; % No Dirichlet
        fprintf('Obtained %i Lagrangian dof of a total of
%i.\n',sum(u0Mask(:,t)>0)*Set.dim,Set.dim*sum(uMask(:,t)>0));
        u0=Mat2Dof(u(:,:,t),u0Mask(:,t));
        fprintf('Applying inverse analysis.\n')
        tMask(:,t)=u0Mask(:,t); % Apply loads only at nodes with high
correlation

[tractions,u(:,:,t),C]=InverseFEM(Set,C,X,u0,uD,eMask,mMask,tMask(:,t),uMask(:
,t));
        fprintf('Sum Ty(+-)=%e, Sum
Ty(+)=%e\n',sum(tractions(tractions(:,2)==2,3)),sum(tractions(tractions(:,2)==
2 & tractions(:,3)>0,3))/length(tractions(tractions(:,2)==2 &
tractions(:,3)>0,3)));
        if Set.Lagrangian
            [uL,u0MaskL(:,t)]=Spat2Lagr(u(:,:,t),x(:,:,t),X);
            x(:,:,t)=x(:,:,t)+uL;
        end
        [tM(:,:,t),tMask(:,t)]=Dof2Mat(Set.dim,Set.nodes,tractions);
    else
        tM(:,:,t)=zeros(size(u(:,:,t)));
    end
    x(:,:,t+1)=x(:,:,t);
    % Write at each increment (get at least partial results)

uMaskL=VTKPIV(name,C,c,cMask,tM,tMask,tmin,t,u,uMask,u0Mask,u0MaskL,x,X,Set.La
grangian);

save(strcat(name,'.mat'),'name','D','t','X','C','ux','uy','uz','c','cMask','tM
ask','uMask','u0Mask','u0MaskL','x','Set','tM','u','tmin','tmax','uMaskL');
end
if sum(uart)>0
   [EDispl]=PIVError(uart,u,uMask,tmax,pSizes,L);
    t=zeros(1,tmax-1);
      for i=1:tmax-1
          t(1,i)=i;
      end
   figure;
```

```
   subplot(2,2,1);stem(t,EDispl(:,1));title('Relative Error X
Displacement');xlabel('time(s)');ylabel('Er(%)');
   subplot(2,2,2);stem(t,EDispl(:,2));title('Relative Error Y
Displacement');xlabel('time(s)');ylabel('Er(%)');
   subplot(2,2,3);stem(t,EDispl(:,3));title('Relative Error Z
Displacement');xlabel('time(s)');ylabel('Er(%)');
end
```

## ParametersArtificialDisplacement

```
%SIZE IMAGE
xpixel=50;
ypixel=50;
zpixel=50;
time=2;
%DIMENSION'S DOMAIN
Lx=30;
Ly=30;
Lz=30;
%STAIN'S DOMAIN
% The stain must be inside the hole image, so the values can't be
% higher than the image dimensions.
% p=[xmin xmax ymin ymax zmin zmax];
p=[5 10 5 10 5 10];
%DISPLACEMENT
ux=20; % X axis displacement
uy=20; % Y axis displacement
uz=0; % Z axis displacement
uart=[ux uy uz];
%DIMENSION'S SUBVOLUME
% sx=32;
% sy=32;
% sz=32;
```